Reinforcement Learning and Decision Making 2015

Edmonton, AB
June 7th - 10th, 2015

Extended Abstracts

June 7 - June 10, 2015
University of Alberta
Edmonton, AB, Canada
www.rldm.org

TABLE OF CONTENTS

Model parsimony and predictive power of computational models of cognition

Tilman Lesch
Department of Psychiatry
University of Cambridge
tcl36@cam.ac.uk

Mike Aitken Deakin
Department of Psychology
King's College London
michael.aitken@kcl.ac.uk

Barbara J Sahakian
Department of Psychiatry
University of Cambridge
bjs1001@medschl.cam.ac.uk

Abstract

The use of model parsimony to select appropriate models for analysing human behaviour can limit the explanatory depth and power of analysis by restricting the number of parameters included in the best-fitting model or the number of individuals included in the analysis. Here we present an extended q-learning model to investigate the effect of payoff framing on counterfactual updating. Parameter recovery is then used to determine whether preferring the simplest, plausible explanation gives the best measurement. The full and the parsimonious model recovered the original parameter values from the simulated data similarly. Although parameter values recovered by the full model were more variable it did not justify using the parsimonious model to investigate individual differences in parameter values estimated from the task behaviour. The present study provides a guideline for how parameter values based on an a priori model can be assessed to justify the use of a full model over a parsimonious.

Keywords: q-learning, model recovery, model parsimony, decision-making, uncertainty

Acknowledgements

This work was kindly funded by a Core Award from the Medical Research Council and the Wellcome Trust to the Behavioural and Clinical Neuroscience Institute (MRC Ref G1000183; WT Ref 093875/Z/10/Z). Tilman Lesch is supported by the Friedrich-Naumann-Foundation and the University of Cambridge.

1 Introduction

Mathematical models are commonly used in cognitive psychology to characterise processes in human behaviour such as behavioural strategies, susceptibility to biases or the role of environmental factors for behaviours [1], [2]. One of the challenges of such modelling is the selection of appropriate models to explain the data being analysed. Meaningful measures and criteria are needed to analyse overall suitability (absolute model fit) and to compare different models (relative model fit). Differentiating between competing models by evaluating their ability to predict behaviour is complicated by the fact that models often differ in a number of ways including number of parameters and complexity. The most commonly used methods in contemporary psychology to assess model fit are goodness of fit (GOF) and model parsimony (corrected GOF). Both GOF and model parsimony are methods of relative model comparison; they compare one model to another – relative model fit – but cannot assess absolute model fit or model validity – whether the model is appropriate to describe the observed data in the first place. Commonly used measures of goodness of fit are mean squared error or maximum likelihood: typical examples of model parsimony – corrected GOF – are the Akaike Information Criterion [3] or the Bayesian Information Criterion [4]. A measure of absolute model fit is model flexibility or generalizability; it describes the ability of a model to make valid predictions about behaviour not only for one task but also for several tasks [6]. A model that has more parameters may provide a better description of the observed data on a particular task; yet fail in terms of its ability to generalize to other tasks.

Some studies test several different models to identify the model, which best predicts the sample behaviour [5], [6]. This approach allows researchers to make inferences from the winning model's characteristics compared to the inferior models. A potential disadvantage of this approach is that the model that best fits to data might not necessarily reflect processes of psychological interest – limiting the ability to investigate group or individual differences based on model parameters. On the other hand, researchers can define an a priori model to characterise theoretical interest, and look at group or individual differences of the model parameters [7], [8]. This approach enables researchers to choose any model of interest to explain the data; however standard tools to assess model fit – GOF and model parsimony – are no longer appropriate as they assess only relative model fit. Therefore it is often unclear the extent to which the a priori model provides a useful description of the processes underlying the data.

This current study used a data set from a behavioural task typical of the application of RL model fitting in mainstream psychology. It sets out to determine whether for the case of nested models the use of the full model can be justified providing reliable parameter estimates for the analyses of group or individual differences using a novel probabilistic learning task. We specify an a-priori nested model to describe all processes of interest, and then compare parameter recovery of the full model GOF with a parsimonious model based on corrected GOF to determine which approach, if either, is preferable to recover true parameter values from simulated data.

2 A probabilistic behavioural task involving counterfactual feedback

There are considerable differences in the way in which positive and negative feedback is perceived, informs learning and leads to feelings of regret and behaviour to avoid potential negative feedback [9]. Counterfactual beliefs regarding "What could have happened" have been shown to strongly impact human behaviour besides the potential feeling of regret [10]. Behavioural data on a novel four-alternative probabilistic learning task were acquired from 83 participants. In the task individuals were instructed to maximize their reward by choosing continuously between four different stimuli. Feedback was presented not only for the selected stimulus but also for the forgone options (counterfactual feedback). The total of 220 trials was divided into three sections differing in the rate at which the probabilities changed between the four stimuli. On trials 1-40 probabilities remained constant; on trial 41 each option was assigned a different probability of reward, which remained constant between trials 41 to 120. On trials 121 through 220 the probabilities changed every 20 trials between the four stimuli. Throughout both the stable and dynamic conditions, the probabilities were programmed such that there was always one stimulus with a high (.85), one low (.15) and two with a 50/50 probability of reward. Each stimulus paid the same reward per trial if chosen and successful. There was no change in total reward for an unsuccessful choice.

3 A q-learning model of counterfactual feedback

To model the behaviour on this task, the classic q-learning model was extended such that after a trial on which stimulus j ε {1,2,3,4} was chosen, and outcomes O_i {1 = success, 0 = failure} revealed, for *each* stimulus i ε {1,2,3,4}, the value V_i was updated in the direction of O_i using a delta rule with Learning Rate α; tendencies of participants' learning to be modulated according to whether a stimulus was selected (*Selection Bias* γ), the outcome of a stimulus (*Outcome Bias* β), and the outcome of the selected stimulus (*Selection-Outcome Bias* δ), was achieved by the use of parameters which modulated the learning on each trial.

Value Function: Expected reward of stimulus i on trial $t+1$: $Q_i(t+1)=\alpha^*\gamma^*\beta^*\delta^*O_i(t)+(1-\alpha^*\gamma^*\beta^*\delta)^*Q_i(t)$

With trial t ε {1,2,…,220}, stimulus i ε {1,2,3,4}; outcome stimulus i on trial t: $O_i(t)$ ε {0,1}; expected value Q_i for option i on trial t: $Q_i(t)$; α always between 0 and 1; γ = 1 if $i=j$ *(i.e.* selected), free otherwise; β=1 if $O_i(t)=1$, free otherwise, and δ=1 if $i=j$ *or* if $i\neq j$ and $O_i(t)=0$ (i.e. selected or did not win), free otherwise. The value of α *(Learning Rate)*, γ (*Selection-Bias*), β (*Outcome-Bias*), and δ (*Selection-Outcome-Bias*) was bound between zero and one.

To model participants' choices based on the fully parameterized model described above a softmax choice rule with one parameter τ (*Exploitation Rate*) for model stickiness was used – how likely individuals follow the model-based reward-maximizing prediction [11]. The probability of choosing stimulus i is increased by increased value, and decreased by increases in the value of other stimulus values.

Choice Function: probability stimulus i chosen on trial $t+1$: $P_i(t+1)=\frac{\exp(Q_i(t)*\tau)}{\sum_{k=1}^{4}\exp(Q_k(t)*\tau)}$

We followed the typical approach to fitting the model for each individual. Values of all five free parameters (*Learning Rate* α, *Selection Bias* γ, *Outcome Bias* β, *Selection-Outcome Bias* δ and *Exploitation Rate* τ) that maximized the likelihood of the behavioural data were determined for each participant separately [12]. Corrected goodness of model fit was assessed using the Bayesian Information Criterion [BIC, 5]. For each participant, a set of model parameters was obtained, derived on the performance on both the first and second run through the task using the full model and Maximum Likelihood Estimation [MLE, 12].

The key question is the degree to which this 'standard' approach may be considered to have correctly characterised the individual variation in the psychological processes of interest. To answer this question, we investigated performance of the modelling procedure under the assumption that it had been successful. 30 sets of pseudo-behavioural data were then generated for each individual participant's set of model parameters. Afterwards a total of eight nested models were fitted to every simulated data set and the parameter values, which maximized the likelihood of the data, were noted (for each of the 30 simulations for each of 83 participants). The eight models are as follows. Model 1 included two free parameters (Learning Rate α and Exploitation Rate τ, original q-learning with δ-rule); Models 2a, 2b and 2c included both parameters from Model 1 and one out of the three bias parameters (γ, β or δ). Models 3a, 3b, and 3c included both parameters from Model 1 and two out of three bias parameters. Model 4 included all parameters (full model); it is the same model that was used to generate the simulated data sets. All models are nested in *Model 4* by setting the appropriate bias parameters equal to one.

If the task and modelling procedure are a reasonable method for estimating an individual's parameters, then the mean for each model parameter across the 30 data sets should reflect (recover) the corresponding parameter used to generate the pseudo-behavioural data. The similarity of the 30-parameter estimates obtained in this way to the parameters *known* to have generated these data sets thus reflect a measure of the suitability of a model-fitting process (typical to contemporary psychology) as a whole to this task.

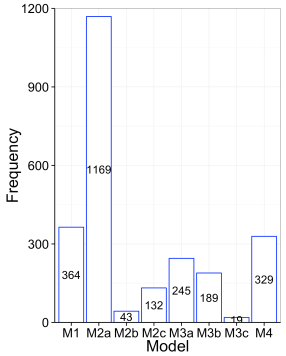


Figure 1: Frequency tables of the most parsimonious model

3 Results

In *Figure 1* the frequency of each model being the best model according to a parsimonious account was plotted across all participants and simulations. For

more than half of the simulated data sets that were generated, the best model according to a parsimony criterion was not the full model that was used to generate the data, but rather Model 2a with three free parameters: Learning Rate α, Exploitation Rate τ and Selection Bias γ.

To further assess parameter recovery the original parameter values and the recovered parameter values for the full model (M4) as well as the original parameter values and the parsimonious model were plotted (Figure 2); the left plot depicts the results for the "Full Model" and right plot for the "Parsimonious Model". On the X-axis the parameter values used to generate the pseudo-behavioural data; black dots represent the recovered parameter value for each of the simulations per individual participant. The black line is the linear regression line with original parameter values as predictor variables and the simulated parameter values across all simulations as predicted values. The red line is the angle bisector in the first quadrant. The graph for the parsimonious model contains only the free parameters of the parsimonious model; parameters not in the parsimonious model or equal to one were excluded. If there were perfect recovery and minimal variance, all parameter values would haven been scattered around the angle bisector (red line, ideal) in the first quadrant.

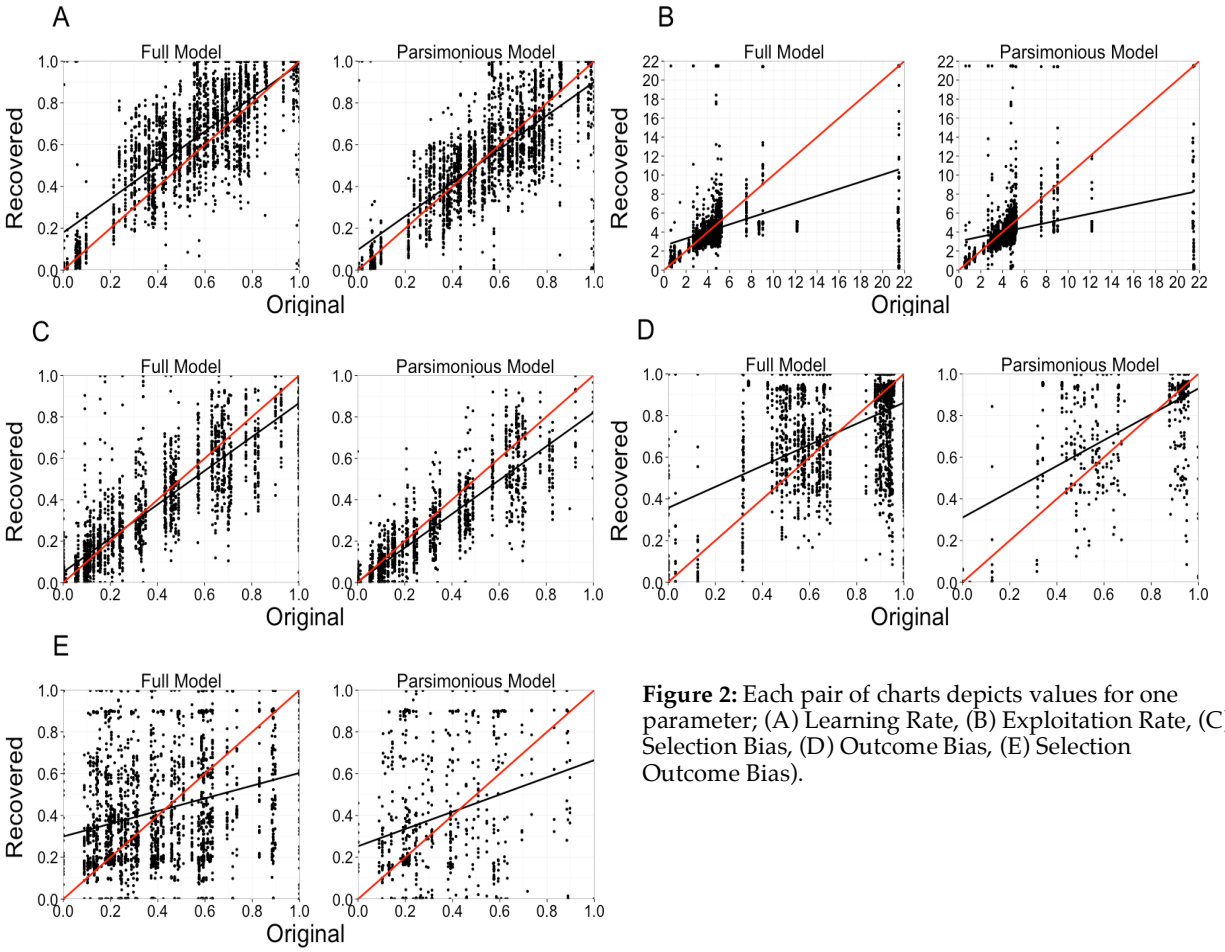


Figure 2: Each pair of charts depicts values for one parameter; (A) Learning Rate, (B) Exploitation Rate, (C) Selection Bias, (D) Outcome Bias, (E) Selection Outcome Bias).

To quantify the discrepancy between parameter values used to simulate the data and parameter estimates recovered from pseudo-behavioural data, the bias and deviation for the full model and parsimonious model were calculated (Table 1). Bias is the mean difference between the recovered parameter estimate and the original parameter value averaged across all simulations. Deviation is the square root of the mean squared difference between the recovered parameter estimates and the original parameter value used to generate the data averaged across all simulations.

Model:	Full Model		Parsimonious	
	Bias	Deviation	Bias	Deviation
Learning Rate α	-.072	5.014	**.013**	**4.485**
Exploitation Rate τ	**.668**	**59.984**	.888	68.513
Selection Bias γ	**.026**	3.618	.045	**2.574**
Outcome Bias β	**.012**	6.274	-.0445	**2.893**
Selection-Outcome Bias δ	**.021**	8.513	-.054	**4.309**

Table 1: Bias and Deviation between original parameter values and recovered parameter values using the best fitting model and the parsimonious model; **BOLD** = minimum value.

For all parameters there was a positive linear trend between the generating parameter and the recovered parameter, confirming that the parameters of the generating model were, to an extent, recoverable by the model fitting used. However, there was considerable variability in the parameter estimates recovered from pseudo-behavioural data for each set of parameters. The positive linear relationship was clearest for Learning Rate α and Selection Bias γ suggesting the most reliable recovery of those parameter values.

The two approaches (full versus parsimonious model estimation) performed in a similar fashion, with bias being smaller for the true model relative to the parsimonious model, and the deviation was smaller for the parsimonious model compared to the full models. Larger variability implies that parameter estimates based on a single behavioural test sessions would be less well able to characterise the processes underlying behaviour of a specific individual. However, while the parsimony-based estimates gave less discrepant recovery of the true parameters, it must be noted that this approach produced no estimates of the final two parameters (β and δ) for over 50% of simulated data sets.

4 Concluding Remarks

The current study used parameter recovery from simulated data to investigate the reliability of simple RL modelling fitting as frequently performed within the mainstream psychology literature. The results showed that using an *a priori* model with high explanatory power – and a larger number of parameters than a more parsimonious model - may be justified where estimates for all parameters are required. Selecting estimates only from the most parsimonious models tended, unsurprisingly, to generate estimates closer to the generating parameter: however, this approach has the disadvantage that many individuals in the sample could not be used for analysis of group or individual differences in model-based parameter values, and there is a suggestion the parsimony restriction could generate a bias in the estimated parameters.

References

[1] J. R. Busemeyer, J. C. Stout, and P. R. Finn, "Using Computational Models to Help Explain Decision Making Processes of Substance Abusers," in *Cognitive and Affective Neuroscience of Psychopathology*, D. Barch, Ed. Oxford University Press, 2003, pp. 1–41.

[2] A. D. Redish, "Addiction as a computational process gone awry," *Science (80-.).*, vol. 306, no. 5703, pp. 1944–7, Dec. 2004.

[3] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automat. Contr.*, vol. 19, no. 6, pp. 716–723, 1974.

[4] G. Schwarz, "Estimating the dimension of a model," *Ann. Stat.*, vol. 6, no. 2, pp. 461–464, 1978.

[5] M. a Pitt, I. J. Myung, and S. Zhang, "Toward a method of selecting among computational models of cognition.," *Psychol. Rev.*, vol. 109, no. 3, pp. 472–491, 2002.

[6] J. R. Busemeyer and J. C. Stout, "A contribution of cognitive decision models to clinical assessment: Decomposing performance on the Bechara gambling task.," *Psychol. Assess.*, vol. 14, no. 3, pp. 253–262, 2002.

[7] E. Yechiam and G. Hochman, "Loss-aversion or loss-attention: the impact of losses on cognitive performance.," *Cogn. Psychol.*, vol. 66, no. 2, pp. 212–31, Mar. 2013.

[8] J. C. Stout, S. L. Rock, M. C. Campbell, J. R. Busemeyer, and P. R. Finn, "Psychological processes underlying risky decisions in drug abusers.," *Psychol. Addict. Behav.*, vol. 19, no. 2, pp. 148–57, Jun. 2005.

[9] E. Yechiam and G. Hochman, "Losses as modulators of attention: review and analysis of the unique effects of losses over gains.," *Psychol. Bull.*, vol. 139, no. 2, pp. 497–518, Mar. 2013.

[10] G. Coricelli, H. D. Critchley, M. Joffily, J. P. O'Doherty, A. Sirigu, and R. J. Dolan, "Regret and its avoidance: a neuroimaging study of choice behavior.," *Nat. Neurosci.*, vol. 8, no. 9, pp. 1255–62, Sep. 2005.

[11] R. D. Luce, *Individual choice behavior*. New York, NY: John Wiley, 1959.

[12] I. J. Myung, "Tutorial on maximum likelihood estimation," *J. Math. Psychol.*, vol. 47, pp. 90–100, 2003.

Bootstrapping Skills

Mankowitz J. Daniel
danielm@tx.technion.ac.il
Technion Israel Institute of Technology Haifa, Israel

Mann A. Timothy *
timothymann@google.com
DeepMind London, UK

Mannor Shie
shie@ee.technion.ac.il
Technion Israel Institute of Technology Haifa, Israel

Abstract

The monolithic approach to policy representation in Markov Decision Processes (MDPs) looks for a single policy that can be represented as a function from states to actions. For the monolithic approach to succeed (and this is not always possible), a complex feature representation is often necessary since the policy is a complex object that has to prescribe what actions to take all over the state space. This is especially true in large-state MDP domains with complicated dynamics. It is also computationally inefficient to both learn and plan in MDPs using a complex monolithic approach. We present a different approach where we restrict the policy space to policies that can be represented as combinations of simpler, parameterized skills—a type of temporally extended action, with a simple policy representation. We introduce Learning Skills via Bootstrapping (LSB) that can use a broad family of Reinforcement Learning (RL) algorithms as a "black box" to iteratively learn parametrized skills. Initially, the learned skills are short-sighted, but each iteration of the algorithm allows the skills to bootstrap off one another, improving each skill in the process. We prove that this bootstrapping process returns a near-optimal policy. Furthermore, our experiments demonstrate that LSB can solve MDPs that, given the same representational power, could not be solved by a monolithic approach. Thus, planning with learned skills results in better policies without requiring complex policy representations.

Keywords: Reinforcement Learning, skills, learning

Acknowledgements

The research leading to these results has received funding from the European Research Council under the European Unions Seventh Framework Program (FP/2007-2013) / ERC Grant Agreement n. 306638.

*This author completed the research for this work while working for the Technion

1 Introduction

State-of-the-art Reinforcement Learning (RL) algorithms need to produce compact solutions to large or continuous state Markov Decision Processes (MDPs), where a solution, called a policy, generates an action when presented with the current state. One such approach to producing compact solutions is linear function approximation.

MDPs are important for both planning and learning in *Reinforcement Learning (RL)*. The RL planning problem uses an MDP model to derive a policy that maximizes the sum of rewards received, while the RL learning problem learns an MDP model from experience (because the MDP model is unknown in advance). In this paper, we focus on RL planning, and use insights from RL that could be used to scale up to problems that are unsolvable with traditional planning approaches (such as Value Iteration and Policy Iteration (c.f., Puterman 1994)). A general result from machine learning is that the sample complexity of learning increases with the complexity of the representation Vapnik & Vapnik (1998). In a planning scenario, increased sample complexity directly translates to an increase in computational complexity. Thus monolithic approaches - a single parametric policy that solves the entire MDP, scale poorly. This is because they often require highly complex feature representations, especially in high-dimensional domains with complicated dynamics, to support near-optimal policies. Instead, we investigate learning a collection of policies over a much simpler feature representation (compact policies) and combine those policies hierarchically.

Generalization, the ability of a system to perform accurately on unseen data, is important for machine learning in general, and can be achieved in this context by restricting the policy space, resulting in compact policies Sutton (1996). Compact policies can be represented and combined hierarchically as *Temporally Extended Actions* (TEAs, Sutton et al., 1999). TEAs are control structures that execute for multiple timesteps. They have been extensively studied under many different names, including skills Konidaris & Barto (2009), macro-actions Hauskrecht et al. (1998); He et al. (2011), and options Sutton et al. (1999). TEAs are known to speed up the convergence rate of some MDP planning algorithms Sutton et al. (1999); Mann & Mannor (2014). Learning a useful set of TEAs has been a topic of intense research McGovern & Barto (2001); Konidaris & Barto (2009); Brunskill & Li (2014). However, prior work suffers from one of the following drawbacks: (1) lack of theoretical analysis guaranteeing that the derived policy will be near-optimal, (2) the process of learning TEAs is so expensive that it needs to be ammortized over a sequence of MDPs, (3) the approach is not applicable to MDPs with large or continuous state-spaces, or (4) the learned TEAs do not generalize over the state-space. In this work, we address these drawbacks in the form of a skill-learning algorithm and a formal theoretical analysis thereof.

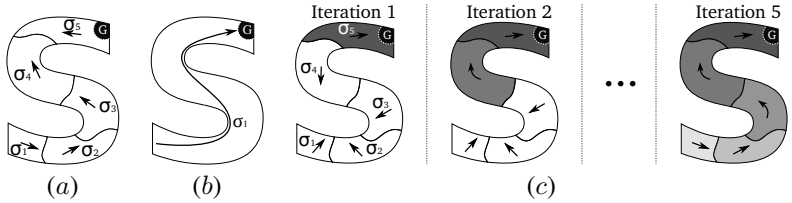


Figure 1: TEAs in an episodic MDP with S-shaped state-space. (a) Although most actions (represented as movements in a single, linear direction) move towards the goal, σ_5 moves away from the goal making it impossible to complete the task. (b) Planning becomes trivial when a single *TEA* (the monolithic approach) takes the agent directly to the goal region. However, the policy is more complex and difficult to learn. (c) Learning skills (denoted by black arrows) in an S-shaped domain with goal region denoted by G. The domain is partitioned into five classes resulting in skill set $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$. In the first iteration, all skills except for σ_5 (which has immediate access to the goal region) are arbitrary. In the second iteration, σ_4 bootstraps off of the reward propagated back by σ_5. This process repeats until useful skills are learned over the entire state-space.

Our main technical contributions are: (1) The introduction of Learning Skills via Bootstrapping (LSB), which requires no additional prior knowledge apart from a partition over the state-space. A high level overview of the algorithm can be seen in Figure 1 c. (2) LSB is the first algorithm for learning skills in continuous state-spaces with theoretical convergence guarantees. (3) Theorem 1, which relates the quality of the policy returned by LSB to the quality of the skills learned by the "black box" RL algorithm. (4) Experiments demonstrating that LSB can solve MDPs that, given the same representational power, can not be solved by a policy derived from a monolithic approach. Thus, planning with learned skills allows us to work with simpler representations Barto et al. (2013), which ultimately allows us to solve larger MDPs.

2 Background

Let $M = \langle S, A, P, R, \gamma \rangle$ be an MDP, where S is a set of states, A is a set of actions, P maps from state-action pairs to probability distributions over next states, R maps each state-action pair to a reward in $[0, 1]$, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi(a|s)$ gives the probability of executing action $a \in A$ from state $s \in S$. The value function of a policy π with respect to a state $s \in S$ is $V_M^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(s_t, a_t) | s_0 = s\right]$ where the expectation is taken with respect to the trajectory produced by following π. We denote the optimal value function by V_M^*. We say that a policy π is ε-optimal if $V_M^\pi(s) \geq V_M^*(s) - \varepsilon$ for all $s \in S$.

One of the key ideas behind skills is that they may be learned locally, but they can be used throughout the entire state-space. We present a new formal definition for skills.

Definition 1. *A* **skill** σ *is defined by a pair* $\langle \pi_\theta, \beta \rangle$*, where* π_θ *is a parametric policy with parameter vector* θ *and* $\beta : S \to \{0, 1\}$ *indicates whether the skill has finished (i.e.,* $\beta(s) = 1$*) or not (i.e.,* $\beta(s) = 0$*) given the current state* $s \in S$*.*

Skills are a special case of the options framework Sutton et al. (1999), and therefore, inherit their theoretical properties (e.g., Precup et al. 1998). However, because skills are defined over parametrized policies, a skill can be initialized anywhere in the state-space. This means that a skill learned in one region of the state-space can potentially be applied in any other region of the state-space.

Definition 2. *Let* Σ *be a set of* $m \geq 1$ *skills. A* **skill policy** μ *is a mapping* $\mu : S \to [m]$ *where* S *is the state-space and* $[m]$ *is the index set over skills.*

A skill policy selects which skill to initialize from the current state by returning the index of one of the skills. By defining skill policies to select an index (rather than the skill itself), we can use the same policy even as the set of skills is adapting. Given a good set of skills, planning can be significantly faster Sutton et al. (1999); Mann & Mannor (2014). However, in many domains we may not be given a good set of skills. Therefore it is necessary to learn this set of skills given the unsatisfactory skill set. In the next section, we introduce an algorithm for dynamically improving skills via bootstrapping.

3 LSB Algorithm

Algorithm 1: **Learning Skills via Bootstrapping (LSB)**

Require: M {Target MDP}, $\mathcal{P}$ {Partitioning of S}, K {# Iterations}

1: $m \leftarrow |\mathcal{P}|$ {# of partition classes}
2: $\mu(s) = \arg\max_{i \in [m]} \mathbb{I}\{s \in \mathcal{P}_i\}$
3: Initialize Σ with m skills. {1 skill per partition.}
4: **for** $k = 1, 2, \ldots, K$ **do** {Do K iterations.}
5: **for** $i = 1, 2, \ldots, m$ **do** {One update per skill.}
6: **Policy Evaluation:**
7: Evaluate μ with Σ to obtain $V_M^{\langle \mu, \Sigma \rangle}$
8: **Skill Update:**
9: Construct Skill MDP M_i' from M & $V_M^{\langle \mu, \Sigma \rangle}$
10: Solve M_i' obtaining policy π_θ
11: $\sigma_i' \leftarrow \langle \pi_\theta, \beta_i \rangle$
12: Replace σ_i in Σ by σ_i'
13: **end for**
14: **end for**
15: **Return** $\langle \mu, \Sigma \rangle$

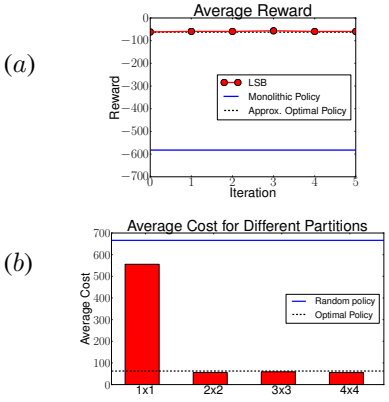
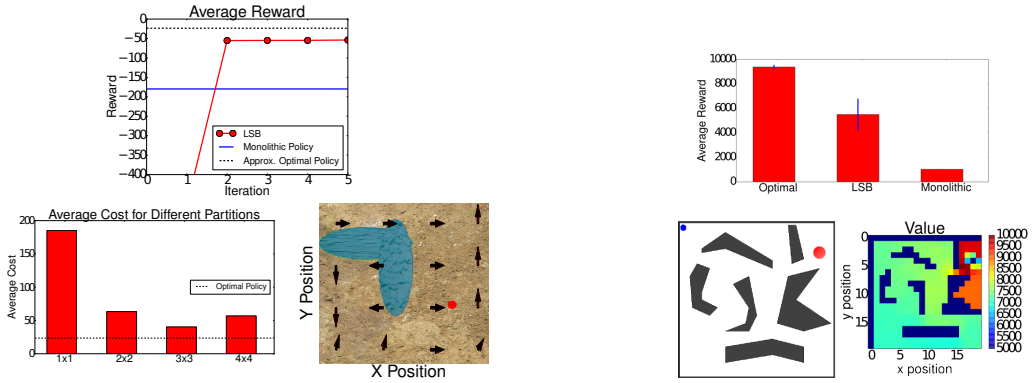


Figure 2: The Mountain Car domain: (a) The average generated by LSB skill policy and the monolithic approach. (b) The average cost (negative reward) for different partitions (i.e., grid sizes).

Learning Skills via Bootstrapping(LSB, Algorithm 1) takes a target MDP M, a partition $\mathcal{P}$ over the state-space and a number of iterations $K \geq 1$ and returns a pair $\langle \mu, \Sigma \rangle$ containing a skill policy μ and a set of skills Σ. The partitioning can be arbitrarily defined but partition classes (sub-partitions within the partitioning) must overlap and the goal region must be inside one of these classes. The number of skills $m = |\mathcal{P}|$ is equal to the number of classes in the partition $\mathcal{P}$ (line 1) as there is one skill per partition class. The skill policy μ returned by LSB is defined (line 2) by $\mu(s) = \arg\max_{i \in [m]} \mathbb{I}\{s \in \mathcal{P}_i\}$, where $\mathbb{I}\{\cdot\}$ is the indicator function returning 1 if its argument is true and 0 otherwise and $\mathcal{P}_i$ denotes the i^{th} class in the partition $\mathcal{P}$. Thus μ simply returns the index of the skill associated with the partition class containing the current state. On line 3, skills are initialized arbitrarily. In our experiments, we initialize the skills with untrained PG algorithms.

Next (lines 4-14), LSB performs K iterations. In each iteration, LSB updates the skills in Σ (lines 5-13). LSB updates each skill individually and then performs policy evaluation in order to ensure convergence. Multiple iterations are needed so that the skill set can converge (Figure 1).

The process of updating a skill (lines 6-12) starts by evaluating μ with the current skill set Σ (line 6). Any number of policy evaluation algorithms could be used here. We used a straighforward variant of LSTD Sorg & Singh (2010). Then we use the target MDP M to construct a Skill MDP M' (line 9). A Skill MDP M_i' is just an episodic MDP that terminates once the agent escapes from $\mathcal{P}_i$ and upon terminating receives a reward equal to the value of the state the agent would have transitioned to in the target MDP. For a formal definition of a Skill MDP, see the supplementary material. Next, LSB uses a planning or RL algorithm to approximately solve the Skill MDP M' returning a parametrized policy π_θ (line 10). Any planning or RL algorithm for regular MDPs could fill this role provided that it produces a parametrized policy. In our experiments, we used a simple actor-critic PG algorithm. Then a new skill $\sigma_i' = \langle \pi_\theta, \beta_i \rangle$ is created (line 11) where π_θ is the policy derived on line 10 and $\beta_i(s) = \begin{cases} 0 & \text{if } s \in \mathcal{P}_i \\ 1 & \text{otherwise} \end{cases}$. The definition of β_i



(a) (b)

Figure 3: Experiments: (a) Puddle World: *Top figure:* The average reward for the LSB algorithm generated by the LSB skill policy, compared to the monolithic approach and an approximately optimal policy derived using Q-learning. *Bottom left:* The average cost (negative reward) for each grid partition. *Bottom right:* Repeatable skills plot. (b) Pinball-world: *Top Figure:* The average reward for the LSB algorithm generated by the LSB skill policy. LSB converges after a single iteration. *Bottom left:* Pinball world. *Bottom right:* Pinball-world value function

means that the skill will terminate only if it leaves the i^{th} partition. Finally, we update the skill set Σ by replacing the i^{th} skill with σ_i' (line 12). Now, we analyze the quality of the policy returned by LSB. It turns out that this depends critically on the quality of the skill learning algorithm.

Definition 3. *Let $\mathcal{P}$ be a partition over the target MDP's state-space. The* **skill learning error** *is defined as $\eta_{\mathcal{P}} = \max_{i \in [m]} \eta_i$, where η_i is the smallest $\eta_i \geq 0$, such that $V^*_{M_i'}(s) - V^{\pi_\theta}_{M_i'}(s) \leq \eta_i$ for all $s \in \mathcal{P}_i$ and π_θ is the policy returned by the skill learning algorithm executed on M_i'.*

The skill learning error quantifies the quality of the Skill MDP solutions returned by our skill learning algorithm. If we used an exact solver to learn skills, then $\eta_{\mathcal{P}} = 0$. However, if we use an approximate solver, then $\eta_{\mathcal{P}}$ will be non-zero and the quality will depend on the partition $\mathcal{P}$. Generally, using finer grain partitions will decrease $\eta_{\mathcal{P}}$. However, the following theorem reveals that adding too many skills can also negatively impact the returned policy's quality.

Theorem 1. *Let $\varepsilon > 0$. If we run LSB with partition $\mathcal{P}$ for $K \geq \log_\gamma (\varepsilon(1-\gamma))$ iterations, then the algorithm returns policy $\varphi = \langle \mu, \Sigma \rangle$ such that*

$$\|V^*_M - V^{\varphi}_M\|_\infty \leq \frac{m\eta_{\mathcal{P}}}{(1-\gamma)^2} + \varepsilon \ , \tag{1}$$

where m is the number of classes in $\mathcal{P}$.

Theorem 1 tells us that when the skill learning error is small, LSB returns a near-optimal policy. The first term on the right hand side of (1) is the approximation error. This is the loss we pay for the parametrized class of policies that we learn skills over. The second term is the convergence error. It goes to 0 as the number of iterations K increases. At first, the guarantee provided by Theorem 1 may appear similar to (Hauskrecht et al. 1998, Theorem 1). However, Hauskrecht et al. 1998 derive TEAs only at the beginning of the learning process and do not update them. On the other hand, LSB updates its skill set dynamically via bootstrapping. Thus, LSB does not require prior knowledge of the optimal value function. Theorem 1 does not explicitly present the effect of policy evaluation error, which occurs with any approximate policy evaluation technique. However, if the policy evaluation error is bounded by $\nu > 0$, then we can simply replace $\eta_{\mathcal{P}}$ in (1) with $(\eta_{\mathcal{P}} + \nu)$. Again, smaller policy evaluation error leads to smaller approximation error.

4 Experiments and Results

We performed experiments on three well-known RL benchmarks: Mountain Car (MC), Puddle World (PW) Sutton (1996) and the Pinball domain Konidaris & Barto (2009). Recall that LSB is a meta-algorithm. We must provide an algorithm for Policy Evaluation (PE) and skill learning. For MC and PW, we used SMDP-LSTD Sorg & Singh (2010) for PE and a modified version of Regular-Gradient ActorCritic Bhatnagar et al. (2009) for skill learning (see supplementary material for details). For Pinball, we use Nearest-Neighbors Function Approximation (NN-FA) for PE and UCB Random Policy Search (UCB-RPS) for skill learning. The purpose of our experiments is to show LSB can solve a complicated task with a simple policy representation by combining bootstrapped skills. In our experiments, each skill is just a probability distribution over actions (independent of the state). We compare their performance to the monolithic approach using the same representation. MC and PW experiments are run for 10 independent trials. Pinball is run for 5 independent trials. A 2×2 grid partitioning is used for the skill partition in MC and PW, whereas a $4 \times 3 \times 1 \times 1$ partitioning is used for pinball.

Figure 2a and Figure 3a *Top* compares the monolithic approach and LSB with a 2×2 grid for MC and PW respectively. In both domains, the monolithic approach achieves low average reward. However, with the same restricted policy representation, LSB

combines skills, resulting in a richer solution space and a higher average reward, as seen in the respective Figures. We then compared the performance on various skill partitions. A 1×1 grid represents the monolithic approach. Partitions of $2 \times 2, 3 \times 3, 4 \times 4$ are also compared. As the size of each partition class decreases (i.e., adding more skills), the skill learning error generally decreases, decreasing the loss as seen in Figures 2b and 3a bottom left respectively. All grid partitions outperform the monolithic approach (1×1 grid). It is also important to note that it is possible for similar skills to be learned in different partition classes. Figure 3a *Bottom right* shows a quiver plot superimposed on a 4×4 grid in PW. For each skill, the direction (black arrows in the figure) is determined by sampling and averaging actions from the skill's probability distribution. Skills in the same direction represent repeatable skills and the skill set of 16 skills can be reduced to a set of 5 skills. Therefore, skill reuse may further reduce the complexity of a solution. Figure 3b *Top* shows the average reward of LSB in the Pinball domain (Figure 3b *Bottom left*). The algorithm outperforms the monolithic approach and converges in a single iteration. The fast convergence is due to NN-FA Policy Evaluation being interlaced between every iteration of skill learning, resulting in fast propagation of the value function (Figure 3b *Bottom right*).

5 Discussion

We introduced a bootstrapping procedure for learning skills. This approach is inspired by, and similar to skill chaining Konidaris & Barto (2009). However, the heuristic approach applied by skill chaining may not produce a near-optimal policy. We provide theoretical results for LSB that directly relate the quality of the final policy to the skill learning error. These are the first theoretical results providing convergence guarantees in a continuous state space using skill learning. One limitation of LSB is that it learns skills for all partition classes. This is a problem in high-dimensional state-spaces, but can be overcome by focusing only on the most important regions of the state-space. One way to achieve this is by observing an expert's demonstrations Argall et al. (2009). One exciting extension of our work would be to incorporate skill interruption, similar to option interruption. Option interruption involves terminating an option based on an adaptive interruption rule Sutton et al. (1999). Options are terminated when the value of continuing the current option is lower than the value of switching to a new option. Mankowitz et al. 2014 have interlaced Sutton's interruption rule between iterations of value iteration and proved convergence to a global optimum. However, their results have not yet been extended to use with function approximation.

References

Argall, Brenna D, Chernova, Sonia, Veloso, Manuela, and Browning, Brett. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. ISSN 0921-8890.

Barto, Andrew, Konidaris, George, and Vigorito, C.M. Behavioral hierarchy: Exploration and representation. In *Computational and Robotic Models of the Hierarchical Organization of Behavior*, pp. 13–46. Springer, 2013.

Bhatnagar, Shalabh, Sutton, Richard S, Ghavamzadeh, Mohammad, and Lee, Mark. Natural actor–critic algorithms. *Automatica*, 45 (11):2471–2482, 2009.

Brunskill, Emma and Li, Lihong. PAC-inspired option discovery in lifelong reinforcement learning. *JMLR*, 1:316–324, 2014.

Hauskrecht, Milos, Meuleau, Nicolas, Kaelbling, Leslie Pack, Dean, Thomas, and Boutilier, Craig. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the 14th Conference on Uncertainty in AI*, pp. 220–229, 1998.

He, Ruijie, Brunskill, Emma, and Roy, Nicholas. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40:523–570, 2011.

Konidaris, George and Barto, Andrew G. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems 22*, pp. 1015–1023, 2009.

Mankowitz, Daniel J, Mann, Timothy A, and Mannor, Shie. Time regularized interrupting options. *ICML*, 2014.

Mann, Timothy A and Mannor, Shie. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *Proceedings of the 31^{st} International Conference on Machine Learning*, 2014.

McGovern, Amy and Barto, Andrew G. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *Proceedings of the 18th International Conference on Machine Learning*, pp. 361 – 368, San Fransisco, USA, 2001.

Precup, Doina, Sutton, Richard S, and Singh, Satinder. Theoretical results on reinforcement learning with temporally abstract options. In *Machine Learning: ECML-98*, pp. 382–393. Springer, 1998.

Puterman, Martin L. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

Sorg, Jonathan and Singh, Satinder. Linear options. In *Proceedings of the* 9^{th} *International Conference on Autonomous Agents and Multiagent Systems*, pp. 31–38. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

Sutton, Richard. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pp. 1038–1044, 1996.

Sutton, Richard S, Precup, Doina, and Singh, Satinder. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, August 1999.

Vapnik, Vladimir Naumovich and Vapnik, Vlamimir. *Statistical learning theory*, volume 2. Wiley New York, 1998.

Covariance Matrix Estimation for Reinforcement Learning

Tomer Lancewicki*
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN 37996
tlancewi@utk.edu

Itamar Arel
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN 37996
itamar@eecs.utk.edu

Abstract

One of the goals in scaling reinforcement learning (RL) pertains to dealing with high-dimensional and continuous state-action spaces. In order to tackle this problem, recent efforts have focused on harnessing well-developed methodologies from statistical learning, estimation theory and empirical inference. A key related challenge is tuning the many parameters and efficiently addressing numerical problems, such that ultimately efficient RL algorithms could be scaled to real-world problem settings. Methods such as Covariance Matrix Adaptation - Evolutionary Strategy (CMAES), Policy Improvement with Path Integral (PI^2) and their variations heavily depends on the covariance matrix of the noisy data observed by the agent. It is well known that covariance matrix estimation is problematic when the number of samples is relatively small compared to the number of variables. One way to tackle this problem is through the use of shrinkage estimators that offer a compromise between the sample covariance matrix and a well-conditioned matrix (also known as the *target*) with the aim of minimizing the mean-squared error (MSE). Recently, it has been shown that a *Multi-Target Shrinkage Estimator* (MTSE) can greatly improve the single-target variation by utilizing several targets simultaneously. Unlike the computationally complex *cross-validation* (CV) procedure, the shrinkage estimators provide an analytical framework which is an attractive alternative to the CV computing procedure. We consider the application of shrinkage estimators in dealing with a function approximation problem, using the *quadratic discriminant analysis* (QDA) technique and show that a two-target shrinkage estimator generates improved performance. The approach paves the way for improved value function estimation in large-scale RL settings, offering higher efficiency and fewer hyper-parameters.

Keywords: covariance matrix estimation, path integral, classification uncertainty

*The authors are with the Machine Intelligence Lab at the University of Tennessee - http://mil.engr.utk.edu

1 Introduction

Reinforcement learning (RL) applied to real-world problems inherently involves combining optimal control theory and dynamic programming methods with learning techniques from statistical estimation theory [1, 2, 3, 4]. The motivation is achieving efficient value function approximation for the non-stationary iterative learning process involved, particularly when the number of state variables exceeds 10 [5]. Recent efforts in scaling RL address continuous state and/or action spaces by optimizing parametrized policies. For example, the *Policy Improvement with Path Integral* (PI^2) [5] combines a derivation from first principles of stochastic optimal control with tools from statistical estimation theory. It has been shown in [6] that PI^2 is a member of a wider family of methods which share probabilistic modeling concepts such as *Covariance Matrix Adaptation - Evolutionary Strategy* (CMAES) [7] and the *Cross-Entropy Methods* (CEM) [8]. The *Path Integral Policy Improvement with Covariance Matrix Adaptation* (PI^2 -CMA) [6] takes advantage on the PI^2 method by determining the magnitude of the exploration noise automatically [6]. The PI^2-SEQ [9] scheme applies PI^2 to sequences of motion primitives. One application of the PI^2-SEQ is concerned with object grasping under uncertainty [9, Sec. 5] while applying the experimental paradigm of [10]. The latter approach has illustrated that over time, humans adapt their reaching motion and grasp to the shape of the object position distribution, determined by the orientation of the main axis of its covariance matrix. Moreover, it has been shown that the PI^2 optimal control policy can be approximated through linear regression [11]. This connection allows the use of well-developed linear regression algorithms for learning the optimal policy. The aforementioned methods rely on accurate covariance matrix estimation of the multivariate data involved. Unfortunately, when the number of observations n is comparable to the number of state variables p the covariance estimation problem become more challenging. In such scenarios, the sample covariance matrix is not well-conditioned and is not necessarily invertible (despite the fact that those two properties are required for most applications). When $n \leq p$, the inversion cannot be computed at all [5, Sec. 2.2].

The same covariance problem arises in other related applications of RL. For example, in RL with Gaussian processes, the covariance matrix is regularized [12, Sec. 2]. However, although the regularization parameter plays a pivotal role, it is not clear how it should be set [12, Sec. 3]. Other related work [13] study the ability to mitigate potentially overconfident classifications by assessing how qualified the system is to make a judgment on the current test datum. It is well known that for a small ratio of training observations n to observation dimensionality p, conventional *Quadratic Discriminant Analysis* (QDA) classifier perform poorly, due to a highly variable class conditional sample covariance matrices. In order to improve the classifiers' performance, regularization is recommended, with the aim of providing an appropriate compromise between the bias and variance of the solution. While other regularization methods [14] define regularization coefficients by the computationally complicated *cross-validation* (CV) procedure, the shrinkage estimators studied in this paper provide an analytical solution, which is an attractive alternative to the CV procedure.

This paper elaborates on the *Multi-Target Shrinkage Estimator* (MTSE) [15] that addresses the problem of covariance matrix estimation when the number of samples is relatively small compared to the number of variables. MTSE offers a compromise between the sample covariance matrix and well-conditioned matrices (also known as *targets*) with the aim of minimizing the mean-squared error (MSE). Section 2 presents the MTSE and examine the squared biases of two diagonal targets. In Section 3, we conduct a careful experimental study and examine the two-target and one-target shrinkage estimator, as well as the *Lediot-Wolf* (LW) [16] method for different covariance matrices. We demonstrate an application for the *quadratic discriminant analysis* (QDA) classifier, showing that the *test classification accuracy rate* (TCAR) is higher when using the two-target, rather than one-target, shrinkage regularization. The QDA classifier is a fundamental component in DeSTIN [17] which is a deep learning system for spatiotemporal feature extraction. The DeSTIN architecture currently assumes diagonal covariance matrices, which is one of the targets examined in this paper. In our future research we intend to utilize the results shown in this paper in order to improve the DeSTIN architecture.

2 Multi-Target Shrinkage Estimation

Let $\{\mathbf{x}_i\}_{i=1}^n$ be a sample of independent identical distributed (i.i.d.) p-dimensional vectors drawn from a density having zero mean and covariance $\mathbf{\Sigma} = \{\sigma_{ij}\}$. The most common estimator of $\mathbf{\Sigma}$ is the sample covariance matrix $\mathbf{S} = \{s_{ij}\}$, defined as

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \tag{1}$$

and is unbiased, i.e., $E\{\mathbf{S}\} = \mathbf{\Sigma}$. The MTSE model [15] defined as

$$\hat{\mathbf{\Sigma}}(\boldsymbol{\gamma}) = \left(1 - \sum_{i=1}^{t} \gamma_i\right)\mathbf{S} + \sum_{i=1}^{t} \gamma_i \mathbf{T}_i, \tag{2}$$

where t is the number of the targets $\mathbf{T}_i$, $i = 1, \ldots, t$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_t]^T$ is the vector of shrinkage coefficients. Our objective is therefore to find $\hat{\boldsymbol{\Sigma}}(\boldsymbol{\gamma})$ (2), which minimizes the MSE loss function

$$L(\boldsymbol{\gamma}) = E\left\{\left\|\hat{\boldsymbol{\Sigma}}(\boldsymbol{\gamma}) - \boldsymbol{\Sigma}\right\|_F^2\right\}. \tag{3}$$

The optimal shrinkage coefficient vector $\boldsymbol{\gamma}$ that minimize $L(\boldsymbol{\gamma})$ (3) can be found by using a strictly convex quadratic program [15]. In this paper, we use the two diagonal targets

$$\mathbf{T}_1 = \frac{\mathrm{Tr}(\mathbf{S})}{p}\mathbf{I}, \qquad \mathbf{T}_2 = \mathrm{diag}(\mathbf{S}). \tag{4}$$

Following the developments in [16, Sec. 2.2], the covariance matrix $\boldsymbol{\Sigma}$ can be written as $\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$, where $\mathbf{V}$ and $\boldsymbol{\Lambda}$ are the eigenvector and eigenvalue matrices of $\boldsymbol{\Sigma}$, respectively. The eigenvalues of $\boldsymbol{\Sigma}$ are denoted as $\zeta_i, i = 1, \ldots, p$ in increasing order, i.e., $\zeta_1 \leq \zeta_2 \leq \ldots \leq \zeta_p$, and it is well known that $\sum_{i=1}^{p} \zeta_i = \mathrm{Tr}(\boldsymbol{\Sigma})$. As a result, the squared bias of $\mathbf{T}_1$ with respect to $\boldsymbol{\Sigma}$ can be written as

$$\|E\{\mathbf{T}_1\} - \boldsymbol{\Sigma}\|_F^2 = \left\|\frac{1}{p}\mathrm{Tr}(\boldsymbol{\Sigma})\mathbf{I} - \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T\right\|_F^2 = \sum_{i=1}^{p}\left(\zeta_i - \bar{\zeta}\right)^2, \qquad \bar{\zeta} = \frac{\mathrm{Tr}(\boldsymbol{\Sigma})}{p} = \frac{1}{p}\sum_{i=1}^{p}\zeta_i \tag{5}$$

where $\bar{\zeta}$ is the mean of the eigenvalues $\zeta_i, i = 1, \ldots, p$. The above result shows that $\|E\{\mathbf{T}_1\} - \boldsymbol{\Sigma}\|_F^2$ is equal to the dispersion of the eigenvalues around their mean. Therefore, $\mathbf{T}_1$ becomes less suitable in describing $\boldsymbol{\Sigma}$ when the dispersion of the eigenvalues (5) increases. On the other hand, the expression of the squared bias of $\mathbf{T}_2$ with respect to $\boldsymbol{\Sigma}$ can be written as

$$\|E\{\mathbf{T}_2\} - \boldsymbol{\Sigma}\|_F^2 = \|\mathrm{diag}(\boldsymbol{\Sigma}) - \boldsymbol{\Sigma}\|_F^2 = \sum_{i \neq j}\sigma_{ij}, \tag{6}$$

which shows that it is equal to the off-diagonal entries in $\boldsymbol{\Sigma}$. Therefore, $\mathbf{T}_2$ becomes less suitable for describing $\boldsymbol{\Sigma}$ when the p variables of $\boldsymbol{\Sigma}$ are more highly correlated.

3 Experiments

In this section, we present an extensive experimental study of one-target and two-target shrinkage estimators. The estimators are affected by the squared bias and the variance of a target, when the latter depends on the number of data observations n. Therefore, we examine cases of different true covariance matrices $\boldsymbol{\Sigma}$ that result in different biases of $\mathbf{T}_1$ and $\mathbf{T}_2$. We then examine the estimator's performance as a function of n. In order to study the effect of the squared biases, we create a $p \times p$ covariance matrix $\boldsymbol{\Sigma}$ with determinant of one, i.e., $|\boldsymbol{\Sigma}| = 1$, according to two parameters. The first parameter is the condition number η, which is the ratio of the largest eigenvalue ζ_{max} to the smallest eigenvalue ζ_{min} of $\boldsymbol{\Sigma}$, i.e., $\eta = \frac{\zeta_{max}}{\zeta_{min}}$. In the experiments, the p eigenvalues of $\boldsymbol{\Sigma}$ denoted as $\zeta_i, i = 1, 2, \ldots, p$ are generated according to

$$\zeta_i = \zeta_{min}\left((\eta - 1)\frac{(i-1)}{(p-1)} + 1\right),\ i = 1, \ldots, p. \tag{7}$$

Then, the eigenvalue matrix $\boldsymbol{\Sigma}$ is defined as having elements $\zeta_i, i = 1, 2, \ldots, p$ in the matrix form

$$\boldsymbol{\Lambda}(\eta) = \mathrm{diag}(\zeta_1, \zeta_2, \ldots, \zeta_p). \tag{8}$$

The second parameter K, controls the rotation of $\boldsymbol{\Lambda}(\eta)$. Our approach is to select a set of orthonormal transformations, as in [18, Sec. 2.B]

$$\mathbf{E}(K) = \prod_{k=1}^{K}\mathbf{E}_k = \mathbf{E}_1\mathbf{E}_2\ldots\mathbf{E}_K, \text{ where each matrix } \mathbf{E}_k \text{is defined as } \mathbf{E}_k = \prod_{l=1}^{p-k}\mathbf{E}_{kl} = \mathbf{E}_{k1}\mathbf{E}_{k2}\ldots\mathbf{E}_{K(p-k)}. \tag{9}$$

The matrix $\mathbf{E}_{kl}$ is an orthonormal rotation of 45^0 in a two-coordinate plane for the coordinates k and $(p + 1 - l)$, i.e.,

$$\mathbf{E}_{kl} = I_{p\times p} + \boldsymbol{\Phi}(k, p + 1 - l), \tag{10}$$

where $\boldsymbol{\Phi}(i_k, j_k)$ is defined as

$$[\boldsymbol{\Phi}]_{ij} = \begin{cases} \frac{1}{\sqrt{2}} - 1 & \text{if } i = j = i_k \text{ or } i = j = j_k \\ \frac{1}{\sqrt{2}} & \text{if } i = i_k \text{ and } j = j_k \\ -\frac{1}{\sqrt{2}} & \text{if } i = j_k \text{ and } j = i_k \\ 0 & otherwise \end{cases}. \tag{11}$$

The parameter K is an integer value with the range $0 \leq K \leq p-1$, where $K = 0$ indicates there is no rotation, and $K = p-1$ indicates full rotation, such that all the coordinates rotate with respect to each other at an angle of 45^0. Then, by using $\mathbf{\Lambda}(\eta)$ (8) and $\mathbf{E}$ (9), the covariance matrix is created by

$$\mathbf{\Sigma}(\eta, K) = \mathbf{E}(K)\,\mathbf{\Lambda}(\eta)\,\mathbf{E}^T(K). \tag{12}$$

By employing the covariance matrix (12), the biases of $\mathbf{T}_1$ and $\mathbf{T}_2$ can be controlled independently for $\eta > 1$. The squared bias $\|E\{\mathbf{T}_1\} - \mathbf{\Sigma}\|_F^2$ is affected only by η, and increases as η does, when $\|E\{\mathbf{T}_1\} - \mathbf{\Sigma}\|_F^2 = 0$ for $\eta = 1$. The $\|E\{\mathbf{T}_2\} - \mathbf{\Sigma}\|_F^2$ is affected only by K, and increases as K does, when $\|E\{\mathbf{T}_2\} - \mathbf{\Sigma}\|_F^2 = 0$ for $K = 0$. It should be noted that if $\eta = 1$ then K has no impact while if η is near 1, then K could has minor impact. The shrinkage estimators used in the study are of the one-target variety with $\mathbf{T}_1$ and $\mathbf{T}_2$. In the figures that appear in this section, these estimators are denoted as T1 and T2, respectively. The LW estimator [16] is of the one-target shrinkage variety with $\mathbf{T}_1$, which uses a biased shrinkage coefficient estimator and is denoted as LW. Finally, the two-target shrinkage estimator appears in the figures as TT. We show that the two-target estimator can improve classification results compared with one-target estimators, when using the *quadratic discriminant analysis* (QDA) method. The purpose of the QDA is to assign observations to one of several $g = 1, \ldots, G$ groups with p-variate normal distributions

$$f_g(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^p\,|\mathbf{\Sigma}_g|}} \exp\left(-0.5\,(\mathbf{x}-\mathbf{m}_g)^T\,\mathbf{\Sigma}_g^{-1}\,(\mathbf{x}-\mathbf{m}_g)\right), \tag{13}$$

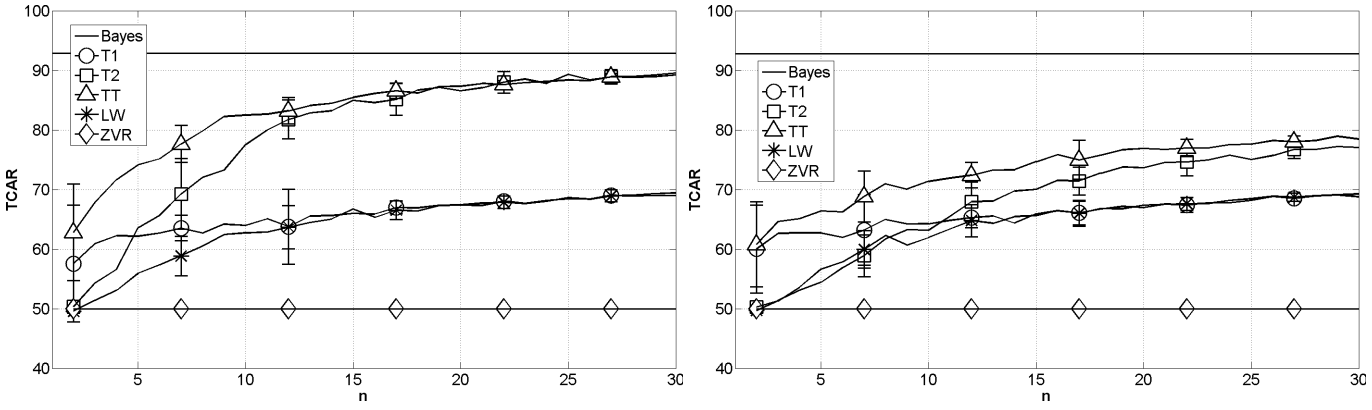
where $\mathbf{m}_g$ and $\mathbf{\Sigma}_g$ are the population mean vector and covariance matrix of the group g. An observation $\mathbf{x}$ is assigned to a class $\hat{g}$ according to

$$d_{\hat{g}}(\mathbf{x}) = \min_{1 \leq g \leq G} d_g(\mathbf{x}), \tag{14}$$

with

$$d_g(\mathbf{x}) = (\mathbf{x}-\mathbf{m}_g)^T\,\mathbf{\Sigma}_g^{-1}\,(\mathbf{x}-\mathbf{m}_g) + \ln|\mathbf{\Sigma}_g| - 2\ln\pi_g, \tag{15}$$

where π_g is the unconditional prior probability of observing a member from the group g. In our experiments, we classify two groups ($G = 2$), with observations generated from a normal distribution with zero mean and $\pi_1 = \pi_2$. The covariance matrix of the first group is the identity matrix $\mathbf{\Sigma}_1 = \mathbf{I}$, while that of the second group is the covariance matrix $\mathbf{\Sigma}_2(\eta, K) = \mathbf{\Sigma}(\eta, K)$ (12), which is generated on the basis of the previous experiments. The goal is to study the effectiveness of the shrinkage estimators when using QDA, by assigning observations to one of these two groups, based on the classification rule (14). We run our experiments for $n = 2, 3, \ldots, 30$. For each n, twenty sets of data of size n are produced.



(a) (b)

Figure 1: QDA for (a) $\mathbf{\Sigma}_2(\eta, 0) = \mathbf{\Lambda}(\eta)$ with $\eta = 10$ and (b) an unrestricted $\mathbf{\Sigma}_2(10, K)$ with $K = 5$

We summarize for each experiment the average *test classification accuracy rate* (TCAR) with standard deviations (the bars in the figure) over the twenty replications for each n. For each group, 10^5 test observations were generated in order to exam the efficiency of the classifier. We provide the best TCAR, calculated by using (14), when the covariance matrices are known, denoted in the figures as Bayes. We also compare the results for a regularization [19, sec. 6], where the zero eigenvalues were replaced with a small number just large enough to permit numerically stable inversion. This has the effect of producing a classification rule based on Euclidean distance in the zero-variance subspace. We denote this procedure as the *zero-variance regularization* (ZVR). In all experiments, the TCAR of the two-target estimator is higher than the one-target variety. The LW estimator is inferior to its unbiased version when dealing with a small number of observations, and converges to its unbiased version as the number of observations increases. Fig. 1(a) presents the result

when the covariance matrix is a diagonal matrix, i.e., $\mathbf{\Sigma}_2(\eta, 0) = \mathbf{\Lambda}(\eta)$, with $\eta = 10$, and therefore $\mathbf{T}_2$ is unbiased while $\mathbf{T}_1$ is biased. The target $\mathbf{T}_1$ provides a higher TCAR than $\mathbf{T}_2$ for small numbers of observations, and then $\mathbf{T}_2$ provides a better TCAR. In Fig. 1(b), the covariance matrix is unrestricted, i.e., $\mathbf{\Sigma}_2(10, K)$, with $K = 5$. The targets $\mathbf{T}_1$ and $\mathbf{T}_2$ are biased. The squared bias of $\mathbf{T}_1$ is not affected by K; whereas the higher the value of K, the higher the squared bias of $\mathbf{T}_2$, and therefore $\mathbf{T}_2$ loses its advantage over $\mathbf{T}_1$.

In conclusion, it has been shown that the *Multi-Target Shrinkage Estimator* (MTSE) [15] can greatly improve the single-target variation in the sense of *mean-squared error* (MSE) by utilizing several targets simultaneously. We consider the application of shrinkage estimator in the context of a function approximation problem, using the *quadratic discriminant analysis* (QDA) technique and show that a two-target shrinkage estimator generates improved performance. This is done by a careful experimental study which examines the squared biases of the two diagonal targets. Unlike the computationally complex *cross-validation* (CV) procedure; the shrinkage estimators provide an analytical solution which is an attractive alternative to the CV computing procedure, commonly used in the QDA. The approach paves the way for improved value function estimation in large-scale RL settings, offering higher efficiency and fewer hyper-parameters.

References

[1] P. Dayan and G. E. Hinton, "Using expectation-maximization for reinforcement learning," *Neural Computation*, vol. 9, no. 2, pp. 271–278, 1997.

[2] M. Ghavamzadeh and Y. Engel, "Bayesian actor-critic algorithms," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 297–304.

[3] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state markov decision processes," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 945–952.

[4] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis, "Learning model-free robot control by a monte carlo em algorithm," *Autonomous Robots*, vol. 27, no. 2, pp. 123–130, 2009.

[5] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, Dec. 2010.

[6] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

[7] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, June 2001.

[8] S. Mannor, R. Y. Rubinstein, and Y. Gat, "The cross entropy method for fast policy search," in *ICML*, 2003, pp. 512–519.

[9] F. Stulp, E. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1360–1370, Dec 2012.

[10] V. N. Christopoulos and P. R. Schrater, "Grasping objects with environmentally induced position uncertainty," *PLoS computational biology*, vol. 5, no. 10, 2009.

[11] F. Farshidian and J. Buchli, "Path integral stochastic optimal control for reinforcement learning," in *The 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM2013)*, 2013.

[12] G. Chowdhary, M. Liu, R. Grande, T. Walsh, J. How, and L. Carin, "Off-policy reinforcement learning with gaussian processes," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 3, pp. 227–238, 2014.

[13] H. Grimmett, R. Paul, R. Triebel, and I. Posner, "Knowing when we don't know: Introspective classification for mission-critical decision making," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, pp. 4531–4538.

[14] P. J. Bickel and E. Levina, "Regularized estimation of large covariance matrices," *The Annals of Statistics*, vol. 36, no. 1, pp. pp. 199–227, 2008.

[15] T. Lancewicki and M. Aladjem, "Multi-target shrinkage estimation for covariance matrices," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6380–6390, Dec 2014.

[16] O. Ledoit and M. Wolf, "A well-conditioned estimator for large-dimensional covariance matrices," *Journal of Multivariate Analysis*, vol. 88, no. 2, pp. 365 – 411, 2004.

[17] S. Young, J. Lu, J. Holleman, and I. Arel, "On the impact of approximate computation in an analog destin architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 934–946, May 2014.

[18] G. Cao, L. Bachega, and C. Bouman, "The sparse matrix transform for covariance estimation and analysis of high dimensional signals," *IEEE Transactions on Image Processing*, vol. 20, no. 3, pp. 625–640, 2011.

[19] J. H. Friedman, "Regularized discriminant analysis," *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175, 1989.

Mitigating Catastrophic Forgetting in Temporal Difference Learning with Function Approximation

Benjamin Goodrich
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN 37996
bgoodric@utk.edu

Itamar Arel
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN 37996
itamar@ieee.org

Abstract

Neural networks have had many great successes in recent years, particularly with the advent of deep learning and many novel training techniques. One issue that has prevented reinforcement learning from taking full advantage of scalable neural networks is that of catastrophic forgetting. The latter affects supervised learning systems when highly correlated input samples are presented, as well as when input patterns are non-stationary. However, most real-world problems are non-stationary in nature, resulting in prolonged periods of time separating inputs drawn from different regions of the input space.

Unfortunately, reinforcement learning presents a worst-case scenario when it comes to precipitating catastrophic forgetting in neural networks. Meaningful training examples are acquired as the agent explores different regions of its state/action space. When the agent is in one such region, only highly correlated samples from that region are typically acquired. Moreover, the regions that the agent is likely to visit will depend on its current policy, suggesting that an agent that has a good policy may avoid exploring particular regions. The confluence of these factors means that without some mitigation techniques, supervised neural networks as function approximation in temporal-difference learning will only be applicable to the simplest test cases.

In this work, we develop a feed forward neural network architecture that mitigates catastrophic forgetting by partitioning the input space in a manner that selectively activates a different subset of hidden neurons for each region of the input space. We demonstrate the effectiveness of the proposed framework on a cart-pole balancing problem for which other neural network architectures exhibit training instability likely due to catastrophic forgetting. We demonstrate that our technique produces better results, particularly with respect to a performance-stability measure.

Keywords: Catastrophic Forgetting, Neural Networks, Reinforcement Learning

1 Introduction

Catastrophic forgetting is a known phenomenon in supervised neural network settings as well as other parameterized learning systems. One issue that prompts the problem of forgetting is correlated samples, which tend to produce gradients of persistent direction. This distorts the parameters of the network, thereby discarding previously learned representations.

In the case of offline training, where one has all of the data available in advance, input examples are usually shuffled. This effectively results in a stationary distribution for the input samples. That is, each example is viewed as drawn independently from a fixed distribution. In online learning, however, one does not have all of the data in advance. The data can only be drawn as it is presented by the environment. If the data is being drawn in a correlated, or non-stationary manner, then this presents major training difficulties for neural networks

With temporal-difference learning in particular, the input distribution is non-stationary for a variety of reasons. Short term correlations occur since each successive state transition will produce a state that is connected to the previous sampled state. This alone can cause problems when training with a neural network approximating the value function, since the network will receive value updates for a sequence of closely related inputs. The input distribution can change over longer time-scales as well since an agent can only be present in one region of the environment at any given time interval, and may spend substantial time in a relatively small number of regions.

This problem is not uncommon since a good policy usually will preclude the agent from visiting certain regions of the state/action space. Consider a near optimal policy for a given MDP. Under many environments, a near optimal policy will dictate that the agent avoid regions that lead to failure. Should a neural network learn an optimal policy, it may quickly forget how to maintain such policy due to lack of visitations at failure regions.

2 Background

Catastrophic forgetting is a well studied phenomenon. In the context of reinforcement learning it has been recently recognized by various researchers, and is at times referred to as the unlearning problem [1] [2] [3].

In a more recent success of applying reinforcement learning combined with deep neural networks, an agent was trained to play Atari video games [4]. One major contribution to the success of this work, was the utilization of a "replay buffer." This replay buffer helps mitigate catastrophic forgetting by storing off-policy experience of the agent and sampling from it randomly when performing updates to the network.

While a replay buffer can somewhat resolve the problem of correlations in training examples, there are issues that it may not be able to overcome. As mentioned in the previous section, an agent that is learning a near-optimal policy may stop visiting certain regions of the state space altogether. If this occurs, then the replay buffer of a fixed size will eventually discard any experience gained from visiting those regions. One could scale a replay buffer by increasing its size at the expense of memory, however scaling in this manner entirely contradicts the very reason neural networks are used to learn in the first place. Neural networks are meant to approximate a mapping without requiring the storing of all the data in the first place. If one has enough memory to store all of the data, then a neural network may not be the best tool for learning. Instead, it may be more appropriate to use other memory-based learning techniques such as localized linear regression [5].

Clearly, catastrophic forgetting mitigation strategies are necessary for facilitating scalability in a more data-driven manner than a replay buffer. Ideally, new neural network architectures should be explored that are inherently designed to address non-stationary input streams, as well as retain representations from prior training trajectories that have not been visited over long periods of time.

3 Neuron Selection Technique

The technique proposed here, dubbed "cluster-select", has recently been introduced as a powerful technique applied to non-stationary classification tasks [6] [7]. In this work, we extend this framework and apply it to online reinforcement learning.

The primary motivation for our work is in the trade-off between techniques that use fully global representations and those that employ strictly local representations. Most neural network activation functions are nonzero for most of the input space, which yield global representations. This means that an update to minimize the error in one region of the input space will affect distant, unrelated regions, likely increasing any error in those regions. Machine learning techniques that employ fully global representations tend to generalize well yet suffer from catastrophic forgetting.

On the other hand, some machine learning techniques employ fully local representations. This would include techniques such as tabular methods, memory-based learning techniques, and radial basis functions. These techniques do not suffer from catastrophic forgetting, however they all have trouble generalizing, because learning something at one region can not be applied to other regions of the input space. Additionally, they all suffer from the curse of dimensionality since for every added dimension, more storage is required. For tabular methods, this manifests as exponentially more memory required with each added dimension. Likewise, radial basis representations will need exponentially more centroids for a representation to effectively cover a high-dimensional input space.

For cluster-select we seek to find a balance between fully global and fully local techniques. There should exist techniques that have some generalization power of global techniques, but also have some properties of local techniques in that they do not suffer from catastrophic forgetting.

With regular feed-forward networks, neuron j can be viewed as having an input weight vector $\vec{w_j}$ associated with it. Cluster-select adds a centroid vector $\vec{c_j}$ in addition to this weight vector. During the feed forward pass, the distance between the input vector $\vec{x}$ and the centroid j is computed using Euclidean norm, such that

$$d_j = \|\vec{x} - c_j\|_2^2 \tag{1}$$

This vector of distances is used to select the k neurons that are nearest to the centroid. Only the k nearest neurons are allowed to have a nonzero output, and all others have their value forced to 0. If an output y_j for a neuron j is allowed to be nonzero, it is computed in the standard way that is done for feed forward neural networks: $y_j = f(\vec{w_j} \cdot \vec{x})$ or simply the dot product of the weights and input (with a bias term assumed to be included as an input), and some nonlinear activation function $f(\cdot)$.

Back-propagation is thus only allowed to occur along the path of the k nearest neurons, with the error derivative forced to 0 for all other neurons. This effectively builds a localized sub-network for each region of the input space, which shares neurons with nearby regions. Effectively, this adds a localized representation to a neural network.

4 Experimental Results

For our test, we considered the the classic cart-pole reinforcement learning problem with no friction. The equations governing the dynamics of this problem can be found in [8]. The problem involves a simulated cart on a horizontal track, with a pole attached to it. The action space has been discretized such that a total of 3 actions involve applying a left force, right force, or no force. This essentially produces bang-bang controls. An episode consists of the cart with the pole started from a random angle, and a small random velocity. An episode proceeds until one of the state variables either grows too large (within reasonable bounds), or 1000 steps elapsed.

Keeping the state variable within reasonable bounds meant that the cart horizontal position and velocity, as well as the pole angular position and velocity were all limited. This was necessary, since a system having no friction would mean that these state variables could be unbounded, potentially producing strange behavior if they grew too large. A negative reward was assigned if the episode ended prematurely due to one of the state variables becoming out of bounds. The goal of the task is to balance the pole upright by applying the horizontal forces to the cart, hence a small positive reward was applied for every frame that the pole was in an upright position.

Each step of the system was simulated using the Runge-Kutta method of numerically solving the differential equations that describe the system. Each step in the simulation consisted of roughly 20.0 milliseconds of simulated time, such that 50 steps equals one second. These tests all used the SARSA learning algorithm with one temporal difference update performed on the network for every step (no batch updates, or replay buffers were used).

To test each method, a random search was performed over hyper-parameters. Hyper-parameters generally included: the learning rate, a small decay constant for the learning rate to decay, the number of hidden neurons, the gamma constant for temporal difference learning, the amount of reward to provide the agent for balancing the pole relative to the amount of negative reward for going out of bounds, the initial ϵ to use for ϵ-greedy exploration, the amount to decay ϵ. For cluster-select, there was an additional hyper-parameter for the number of neurons to select for a feed forward pass.

Each activation function was examined separately, where reasonable selections for the hyper-parameters were provided for the random search. Upon performing approximately 200 runs for each activation function with a given set of hyper-parameters, those that produced the best results were selected. Note that in the plots, performance was measured as a function of the total number of steps that the agent was able to balance the pole and collect reward for an episode. The total reward was not plotted, because the amount of reward to assign was a hyper-parameter.

Figure 1 provides results for a simple case with tabular value function. This particular result had its hyper-parameters hand-tuned (i.e. no random searches of hyper-parameters were performed), and it is provided as a simple baseline level of performance for the tabular case. The value function was maintained in a table of 80,000 states where the entry in the

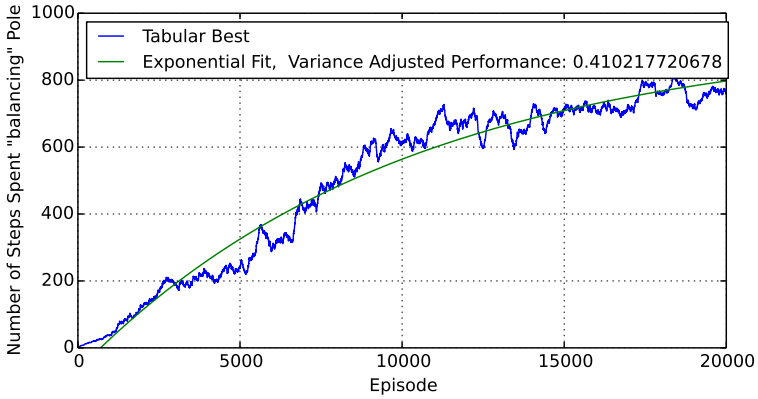


Figure 1: Result for a Tabular $Q_{s,a}$ Estimator

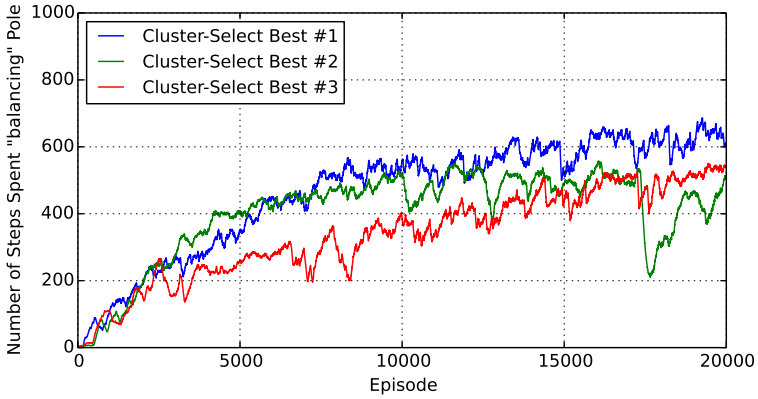


Figure 2: Result for Cluster-Select Neural Net $Q_{s,a}$ Estimator

table was obtained as a function of the 4 state variables. This was done by binning the state variables such that the cart position had 10 bins; the cart velocity, pole angle, and pole angular velocity all had 20 bins. The binning was performed over the valid ranges of these state variables. The number of bins for each state variable was a hyper-parameter which we hand-tuned. Figure 1 also shows a fit to an exponential curve of the form $f(x) = a - b\exp(-cx)$ where a, b, c are constants pertaining to the fitted curve.

Figure 3 illustrates the results for a neural network with linear rectified activation functions. This activation function produced some agents with the best performance. Unfortunately the good performance was unstable, and would often regress as shown in this figure. These agents would learn to balance the pole well, then suddenly regress to terrible performance. We hypothesize that this sudden regression is caused by catastrophic forgetting in the hidden layers. Essentially, after the agent begins to learn to balance the pole well, it is unable to maintain this policy since the network is no longer being trained on the failure states. Eventually it drops the pole, and 'unlearns' the previous captured representation. A plot of performance for networks with sigmoid and hyperbolic tangent activations is not provided, since these networks did not reach adequate levels of performance. It is unclear why these particular activation functions failed to deliver a proficient policy. It is possible that they simply required more training time, or that a good set of hyper-parameters was never found. It is also possible that these activation functions are a poor match for this particular problem.

On the other hand, the cluster-select technique generally had a much smoother learning curve. In particular, the learning profile does not exhibit sudden dips (regressions) in performance, as Figure 2 clearly illustrates. In addition, Table 1 provides an objective measure of performance expressed as the log of the variance-adjusted performance. To compute the latter we first fit the performance curve to an exponential function, as depicted in Figure 1. Next, we measure the mean squared deviation of the original learning curve from the fitted function. Finally, we define the variance-adjusted performance as the mean of the squared values of the original learning curve relative to the mean squared deviation from the fitted function. This metric favors a learner that is both stable in its learning profile as well as reaches a high performance level.

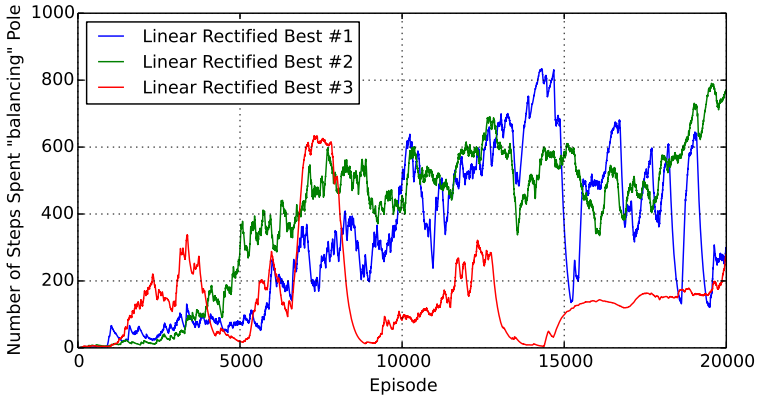


Figure 3: Result for a Linear Rectified Neural Net $Q_{s,a}$ Estimator

	Tabular	Hyperbolic Tangent	Sigmoid	Linear Rectified	Cluster-Select
Performance (Mean Squared Sum)	310311	3674	17682	202616	243417
Deviation from Exponential Fit	1944	225.4	1083	7618	801.6
Log Variance Adjusted Performance	5.073	2.791	2.793	3.281	**5.716**

Table 1: Summary of Results

5 Conclusion

This work has demonstrated an unlearning effect that neural networks exhibit when combined with reinforcement learning. This was illustrated by establishing a test case that exhibits this phenomenon. A neural network architecture was proposed that aims to reduce catastrophic forgetting by localizing activations of the network to regions of the input space. This proposed architecture was shown to substantially reduce this unlearning effect.

We hope that the broader impact of this work will bring attention to this problem of catastrophic forgetting, as more work needs to be put into developing solutions in order for the reinforcement learning community to benefit from modern advances in neural networks and deep learning.

References

[1] V. U. Cetina, "Multilayer perceptrons with radial basis functions as value functions in reinforcement learning." in *ESANN*, 2008, pp. 161–166.

[2] S. Weaver, L. Baird, and M. Polycarpou, "Preventing unlearning during online training of feedforward networks," in *Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings*. IEEE, 1998, pp. 359–364.

[3] J. R. N. Forbes, "Reinforcement learning for autonomous vehicles," Ph.D. dissertation, UNIVERSITY of CALIFORNIA at BERKELEY, 2002.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[5] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema, "Efficient model learning methods for actor–critic control," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 3, pp. 591–602, 2012.

[6] B. Goodrich and I. Arel, "Unsupervised neuron selection for mitigating catastrophic forgetting in neural networks," in *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*. IEEE, 2014.

[7] ——, "Neuron clustering for mitigating catastrophic forgetting in feedforward neural networks," in *Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014 IEEE Symposium on*. IEEE, 2014, pp. 62–68.

[8] R. V. Florian, "Correct equations for the dynamics of the cart-pole system," *Center for Cognitive and Neural Studies (Coneural), Romania*, 2007.

Bootstrapped Linear Bandits

Nandan Sudarsanam
Department of Management Studies
Indian Institute of Technology - Madras
Chennai, Tamil Nadu, India
nandan@iitm.ac.in

Avijit Saha
Computer Science and Engineering
Indian Institute of Technology – Madras
Chennai, Tamil Nadu, India
avijit@cse.iitm.ac.in

Balaraman Ravindran
Computer Science and Engineering
Indian Institute of Technology – Madras
Chennai, Tamil Nadu, India
ravi@cse.iitm.ac.in

Abstract

Keywords: Bootstrap, Linear Stochastic Bandits, Linear Parameterized Bandits, Linear Bandits, Hierarchical Probability Model

This study presents two new algorithms for solving linear stochastic bandit problems. The proposed methods are inspired by the bootstrap approach to create confidence bounds, and therefore make no assumptions about the distribution of noise in the underlying system. We present the *X*-Fixed and *X*-Random bootstrap bandits which correspond to the two well-known approaches for conducting bootstraps on models, in the statistics literature. The proposed methods are compared to other popular solutions for linear stochastic bandit problems such as OFUL [5], LinUCB [6] and Thompson Sampling [9]. The comparisons are carried out using a simulation study on a hierarchical probability meta-model, built from published data of experiments, which were run on real systems. The response surfaces are presented with varying degrees of Gaussian noise for the simulations. The proposed methods perform better than the comparisons, asymptotically, while OFUL and LinUCB perform better in the early stages. The *X*-Random bootstrap performs substantially worse when compared to the *X*-Fixed and the other methods in the initial stages. The proposed methods also perform comparably to a parametric approach which has the knowledge that the noise is Gaussian, both asymptotically and during the early stages. We conclude that the *X*-Fixed bootstrap bandit could be a preferred alternative for solving linear bandit problems, especially when we are dealing with an unknown distribution of the noise. More broadly, this research hopes to motivate the more extended use of non-parametric tools and techniques from statistics to analyse bandit problems.

Acknowledgements

This work was supported through funding from IIT-Madras, CSE/14-15/831/RFTP/BRAV. We also thank Professor Aditya Gopalan for his insights on our work.

1 Introduction

In its classical form, the multi-armed bandit problem requires that a learning agent makes a choice, or selects an action, from n alternatives, across t trials, for each trial. After a choice is made in a given trial, the system presents the learning agent with a numerical reward from a stationary probability distribution associated with the action taken. The goal of the learning agent is to dynamically and sequentially choose alternatives, referred to as arms, which will maximize the expected total reward across the t trials [1]. An extended version of this problem statement is the linear stochastic bandit problem, also referred to as linear bandits, or linear parameterized bandits. Originally proposed by [2] (a variant of the problem was considered by [7] prior to that), this conception varies from the standard multi-armed bandit, in that each arm or alternative is first parameterized into a set of features, which is known to the learning agent. The reward is still a sample from a stationary probability distribution, but its expected value is seen as the inner product of the feature vector and a fixed weight vector, which indicates the influence of each feature on the expected reward. The learning agent therefore seeks to understand these weights in its exploration phase, and uses this to exploit the system by picking promising arms. The critical advantage of this approach is that since we are tagging the rewards to the features, and not the arms directly, we don't need to pull each arm (the set of which could be large or even infinite), but would still be able to learn about the expected reward of each arm by understanding them through a common set features (which is expected be smaller in size). [2,3,4,5, and 6]

The linear stochastic bandit conception allows us to extend various real-world online learning problems to a bandit framework. Studies [7], [2] and [6] discuss the application of a selection problem in displaying internet banner Ads or News articles. The agent needs to choose an Ad among many, to display to a specific user. Here the agent focuses its learning on understanding the features, which are combination of the User's and Ad's characteristics, as opposed to learning directly about each individual ad. The reward is received when a user clicks on the Ad. Similarly, [4] presents an application in marketing where the agent is tasked with choosing a product (arm) to offer to a customer, and the various product characteristics such as price, popularity, etc., are the features. The scope for application also goes beyond selection problems that span single objects or single processes conceptualized as arms (products, Ads). Situations exist where we have a set of decisions, or a combination of actions that need to be conceptually repackaged as a single arm. As discussed in [3], take for instance, the standard case of choosing one out of K clinical treatments, which is modelled as a classical bandit problem with K arms. If we wanted to extend this context to a decision problem where we could choose any combination of the K clinical treatments to be applied on a subject, then the set of possible decisions to choose from increase to 2^k. In this situation, we could treat each of the 2^k combinations as separate arms in a linear bandit framework. We could conceptualize, at minimum, K features (corresponding to presence of absence of a treatment), and perhaps even some interaction terms and higher order transformations as additional features. Another novel application related to such combinatorics is discussed by [8] where an agent needs to figure out, online, the quickest path of getting from one point to another in a network. Here, multiple paths involving a combination of edges in the network can be selected to achieve this task and the agent learns the cost of choosing a certain path, but not the cost of each of the edges. Modelling this through linear bandit framework involves considering each path as an arm, and each edge as a feature.

In this study, we propose two algorithms to solve the linear stochastic bandit problems without making any assumptions about the noise in the system. Inspired by practices in the statistics community to generate confidence bands on linear models through bootstraps [10], we apply these bootstrap algorithms in conjunction with our linear bandit formulation to create an arm pulling agent. Both these algorithms rely on the popular concept of using an upper confidence bound associated with arms [2], derived from the bootstraps, in order to choose an arm. We compare these algorithms to other well researched solutions in the linear bandit space. Specifically, we look at the algorithm OFUL[5], LinUCB[6], and Thompson sampling for linear bandits[9]. In addition to these methods we also present a comparison to a theoretical approach of using confidence bands assuming a Gaussian error. We perform this comparison using a simulations on a hierarchical probability meta-model, which describe response surfaces inspired by real world systems. Section 2 describes the proposed bootstrap algorithms, and in section 3 we describe the test environment, and finally in section 4 we discuss the results with some directions for future research.

2 Algorithms

In a bandit framework, the use of sampling directly from past data to create an arm pulling agent is rarely seen (Thompson sampling, for instance, samples from a distribution, which is created from the data). One notable exception

can be seen with [17]. The bootstrap approaches sample from the data with replacement. The two main algorithms presented in this study are the *X*-Fixed and *X*-random Bootstrap Bandits, corresponding to the similarly named approaches of bootstrapping to evaluate the confidence bands of linear regression models discussed in the statistics literature. The use of these bootstrap approaches, especially with the above mentioned terminology, is first seen in [10] and well discussed in various sources such as [11],[12], and [13]. There are various advantages to using the bootstrap approaches. The main advantage being that these methods allow us to get confidence bounds without making any assumptions on the distributions or independence of variates, unlike some of the comparison methods. While such an approach could be computationally intensive, it could be very helpful in cases where the sampling distribution is unknown, or difficult to derive.

The bootstrap approaches exploit the central idea that '*The population is to the sample, as the sample is to the bootstrap sample*'. Using this principle, the bootstrap can serve as an approach to mimic the variability we might see in a parameter (in our case, the parameters are the mean reward for each arm), by presenting the system with multiple samples. In the first case, *X*-fixed, we fit a linear model to the historic data (step 3), then we bootstrap on the residuals of a fitted model (steps 4-6). We then apply the bootstrapped residuals to the fitted model to create new outputs, and re-regress the new outputs to the fixed inputs (step 7). We repeat these steps multiple times to get multiple linear fits. In the case of *X*-Random we bootstrap each data point, input and output pairs (steps 4-5), and form a regression of the bootstrapped sample (step 6). If we believe that the inputs are fixed, and the output is a random sample from a distribution that is determined by X, then the *X*-Fixed is appropriate. However, if we believe that the inputs characterized by the X matrix is in itself a random sample, then we would find the *X*-Random to be appropriate. The bandit algorithm which encompasses the two regression bootstrap approaches is discussed below:

Assume a system where the full set of all possible unique arms we are interested in is M, and there are F features that describe these arms. In this context we are interested in two different sets of arms, represented by two different matrices. We define matrix U which is a unique set of all the arms, this matrix is of dimension $M \times F$. We also define matrix X which represents the sequence of arms that have been tried, and for which rewards have been gathered. At trial or round t this matrix is of dimension $t \times F$. We present the vector of rewards as vector R which is of also of size t and corresponds to the rewards received from the each row of matrix X.

Algorithm 1: *X*-Fixed Bootstrap Linear Bandit	Algorithm 2: *X*-Random Bootstrap Linear Bandit
1. [1]Run initial seed X and obtain R	1. [1]Run initial seed X and obtain R
2. For each trial t = 1..T	2. For each trial t = 1..T
3. Compute $\beta^* = (X'X)^{-1}X'R$	3. For each Bootstrap b = 1..B
4. Compute $e^* = R - X'\beta^*$	4. Create X_b a bootstrapped sample of X and of the same size as X
5. For each Bootstrap b = 1..B	5. Create R_b from R by matching the selection of data points in X_b
6. Create e_b, a bootstrapped sample of e^* and of the same size as e^*	6. Compute $\beta_b = (X_b{}'X_b)^{-1}X_b{}'(R_b)$
7. Compute $\beta_b = (X'X)^{-1}X'(X'\beta^* + e_b)$	7. For each arm m = 1..M
8. For each arm m = 1..M	8. Compute the performance of the arm defined by the vector $U_{m.}$ across the bootstrap generated betas β_b by defining $Y_{m,b} = U_{m.} \times \beta_b$
9. Compute the performance of the arm defined by the vector $U_{m.}$ across the bootstrap generated betas β_b by defining $Y_{m,b} = U_{m.} \times \beta_b$	9. End arm loop; End bootstrap loop
10. End arm loop; End bootstrap loop	10. Select 95% confidence bound for each arm across the bootstraps $\widehat{Y_m}$=95th percentile(Y_m) across the B bootstraps
11. Select 95% confidence bound for each arm across the bootstraps $\widehat{Y_m}$=95th percentile(Y_m) across the B bootstraps	11. Select arm $U_m^{max} = \max_{m \epsilon M}(\widehat{Y_m})$
12. Select arm $U_m^{max} = \max_{m \epsilon M}(\widehat{Y_m})$	12. Receive reward r associated with arm $U_{m.}$
13. Receive reward r associated with arm $U_{m.}$	13. Update $X = [X;\ U_m^{max}]$, update $R = [R;\ r]$
14. Update $X = [X;\ U_m^{max}]$, update $R = [R;\ r]$	14. End loop for trial
15. End loop for trial	

[1]The algorithms discussed and the comparisons are assumed to be presented with a bare minimum seed of arms and rewards which allow the agent to make an initial estimate for all the f features' weights.

In addition to the above mentioned two algorithms, we also evaluate a parametric approach which conceptualizes the pure noise to be from a Gaussian distribution with mean 0. For this approach we evaluate the upper confidence band (a variant of the confidence bound for functional forms):

$$\widehat{Y_m} = U_m{}'(X'X)^{-1}X'R + t_{0.05,n-2}s\{\tfrac{1}{n} + U_m{}'[X'X]^{-1}U_m\}^{1/2} \quad (1)$$

, where t implies the student–t distribution, and the arm is chosen in accordance to $U_m^{max} = \max_{m \epsilon M}(\widehat{Y_m})$

We compare the bootstrapped linear bandits to parametric Gaussian bound discussed above, and three other established methods of working with linear bandits. These are algorithms OFUL[5], LinUCB[6], and Thompson sampling for linear bandits[9]. The Thompson sampling approach is a Bayesian set up, where, in each step a parameter vector is sampled from a posterior distribution. The arm that maximizes the reward for this sampled vector is chosen, and a corresponding reward is received. The posterior distribution is then updated to account for the newly received reward. On the other hand, with LinUCB and OFUL, in each iteration the parameter vector is calculated through a linear regression. These methods then select the best arm by choosing the one with the highest upper confidence bound.

3 Test environments

Inspired by the examples discussed in [3] and [8], we create a combinatorial application of linear bandits, and perform a simulation study on it. Specifically we replicate the extension of experimenting with clinical treatments. We take a sample case where $K = 7$ treatments can be offered to patients, but these treatments are not mutually exclusive. Hence leading to $2^k = 128$ possible combinations or arms to decide from. We parameterize a total of 28 features for the linear bandits, corresponding to the presence and absence of the 7 treatments, and the $7C_2 = 21$ two-way interactions. Given this feature set, each bandit algorithm is seeded with an initial experimentation from a 2^{7-2}, 32 run factorial experiment, the minimum balanced design required to estimate all the 28 features.

The response surfaces for experimentation are simulated based off of the general linear model, with main, effects, two-way interactions, three-way interactions, no higher order effects, and pure Gaussian noise, as shown in the equation below.

$$y_r(x_1, x_2, .., x_7) = \beta_0 + \sum_{i=1}^{7} \beta_i x_i + \sum_{i=1}^{6} \sum_{j=i+1}^{7} \beta_{ij} x_i x_j + \sum_{i=1}^{5} \sum_{j=i+1}^{6} \sum_{k=j+1}^{7} \beta_{ijk} x_i x_j x_k + \varepsilon \quad (2)$$
$$\text{where} \quad \epsilon \sim N(0, {\sigma_\varepsilon}^2).$$

The selection of response surfaces which could contain three-way interactions was intentionally included despite the bandit parameterizations ending with two-way interactions, to reflect the likely scenarios of missing features in any parameterization exercise. We could think of these as hidden features. In order to make the simulated responses more relevant to real-world response functions, we use a hierarchical probability meta-model (HPM), which was originally proposed in [12]. The HPM provides the mathematical structure to determine βs for the GLM seen in equation (2). A study by [14] populate this model's parameters based off of a meta study of 113 data sets from published studies of different engineering systems. The use of this HPM to evaluate experimental algorithms has been used in other studies by [15, 16]. We are, therefore, dealing with meta-data, and real experiments which inspires our study to an environment of only 128 arms, as opposed to larger set of arms seen in other studies which use purely synthetic data.

4 Results and Discussion

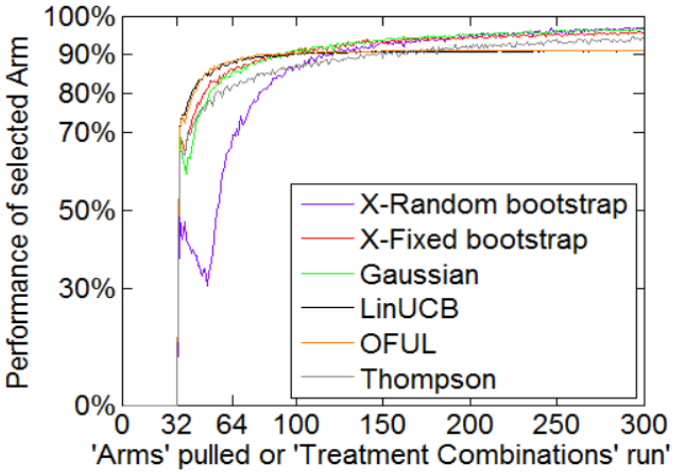


Figure 1: Simulation for $\boldsymbol{\sigma_\varepsilon = 5}$

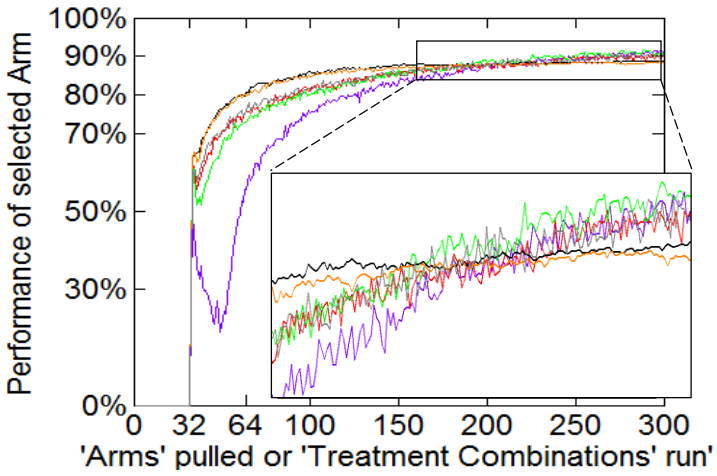


Figure 2: Simulation for $\boldsymbol{\sigma_\varepsilon = 10}$

In this study we look at a simulation of 10,000 response surfaces characterized by versions of equation 2 that have different βs. The pseudo-performance of the selected arm (in line with pseudo-regret discussed in [5]) is described by

$(U_m.\theta/U_m^*.\theta).100$ where U_m is the selected arm which is represented as a vector of its true features, U_m^* is the optimal arm and the true parameters are θ (which are actually unknown to the agents).
We report our findings across both, the $\sigma_\varepsilon = 5$ and the $\sigma_\varepsilon = 10$ scenarios, which correspond to medium and high levels of noise. For instance, the largest standard deviation of the distributions used for generating parameters in equation (2) is equal to 10. The overarching findings of our study is that the proposed algorithms, both the bootstraps, perform better than the comparisons, asymptotically, while OFUL and LinUCB perform best in the early stages. This is despite the fact that three comparisons of OFUL, LinUCB, and Thompson sampling have had their parameters fine-tuned to provide the best performance for the test case, whereas the bootstrap methods, and the Gaussian UCB comparison use the arbitrary, but popularly used bound of the 95th percentile. The Gaussian UCB comparison also has a significant edge, since the Gaussian assumption happens to be replicated by the test environment. The *X*-Random bootstrap works poorly when compared to the *X*-Fixed and most other methods in the initial stages. This should be expected, and is because the initial distribution of X is a careful selection of equi-spaced points from a designed experiment (which is in line with the assumptions of X-Fixed) and not a random sample from the population of X. However, the fact that only a partial set of the features are allowed to be captured could favour the *X*-Random in the long run. Our future efforts seek to create analytical regret bounds for the linear bootstrap approaches, and look at efficient implementations of making the bootstrapped algorithms faster.

5 References

[1] Sutton, R.S. & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT press.

[2] Auer, P. (2003). Using confidence bounds for exploitation-exploration trade-offs. The Journal of Machine Learning Research, 3, 397-422.

[3] Dani, V., Hayes, T. P., & Kakade, S. M. (2008, July). Stochastic Linear Optimization under Bandit Feedback. In COLT (pp. 355-366).

[4] Rusmevichientong, P., & Tsitsiklis, J. N. (2010). Linearly parameterized bandits. Mathematics of Operations Research, 35(2), 395-41

[5] Abbasi-Yadkori, Y., Pál, D., & Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In Advances in Neural Information Processing Systems (pp. 2312-2320).

[6] Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010, April). A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th international conference on World wide web (pp. 661-670). ACM.

[7] Abe, N., & Long, P. M. (1999, June). Associative reinforcement learning using linear probabilistic concepts. In *ICML* (pp. 3-11).

[8] Awerbuch, B., & Kleinberg, R. D. (2004, June). Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (pp. 45-53). ACM.

[9] Agrawal, S., & Goyal, N. (2012). Thompson sampling for contextual bandits with linear payoffs. CoRR,vol.abs/1209.3352, 2012

[10] Thompson, M. L. (1978). Selection of variables in multiple regression: Part I. A review and evaluation. International Statistical Review/Revue Internationale de Statistique, 1-19.

[11] Efron, B., & Tibshirani, R. J. (1994). An introduction to the bootstrap. CRC press.

[12] Breiman, L. (1992). The little bootstrap and other methods for dimensionality selection in regression: X-fixed prediction error. *Journal of the American Statistical Association*, 87(419), 738-754.

[13] Breiman, L., & Spector, P. (1992). Submodel selection and evaluation in regression. The X-random case. *International statistical review/revue internationale de Statistique*, 291-319.

[14] Li, X., Sudarsanam, N., & Frey, D. D. (2006). Regularities in data from factorial experiments. Complexity, 11(5), 32-45.

[15] Frey, D. D., & Li, X. (2008). Using hierarchical probability models to evaluate robust parameter design methods. Journal of Quality Technology, 40(1), 59.

[16] Sudarsanam, N., & Frey, D. D. (2011). Using ensemble techniques to advance adaptive one-factor-at-a-time experimentation. Quality and Reliability Engineering International, 27(7), 947-957.

[17] Baransi, A., Maillard, O. A., & Mannor, S. (2014). Sub-sampling for Multi-armed Bandits. In Machine Learning and Knowledge Discovery in Databases (pp. 115-131). Springer Berlin Heidelberg.

Model-based strategy selection learning

Falk Lieder
Helen Wills Neuroscience Institute
UC Berkeley
Berkeley, CA
falk.lieder@berkeley.edu

Thomas L. Griffiths
Department of Psychology
UC Berkeley
Berkeley, CA
tom_griffiths@berkeley.edu

Abstract

Humans possess a repertoire of decision strategies. This raises the question how we decide how to decide. Behavioral experiments suggest that the answer includes metacognitive reinforcement learning: rewards reinforce not only our behavior but also the cognitive processes that lead to it. Previous theories of strategy selection, namely SSL and RELACS, assumed that model-free reinforcement learning identifies the cognitive strategy that works best *on average* across all problems in the environment. Here we explore the alternative: model-based reinforcement learning about how the differential effectiveness of cognitive strategies depends on the features of individual problems. Our theory posits that people learn a predictive model of each strategy's accuracy and execution time and choose strategies according to their predicted speed-accuracy tradeoff for the problem to be solved. We evaluate our theory against previous accounts by fitting published data on multi-attribute decision making, conducting a novel experiment, and demonstrating that our theory can account for people's adaptive flexibility in risky choice. We find that while SSL and RELACS are sufficient to explain people's ability to adapt to a homogeneous environment in which all decision problems are of the same type, model-based strategy selection learning can also explain people's ability to adapt to heterogeneous environments and flexibly switch to a different decision-strategy when the situation changes.

Keywords: Strategy Selection; Decision-Making; Heuristics; Bounded Rationality; Cognitive Control; Learning

Acknowledgements

This work was supported by ONR MURI N00014-13-1-0341 and grant number N00014-13-1-0341 from the Office of Naval Research. The abstract is based on Lieder and Griffiths (2015).

1 Introduction

How should we decide? There is no single best decision strategy because different problems require different tools. Fortunately, the mind appears to be equipped with a toolbox of different strategies each of which is adaptive for a particular set of problems (Payne, Bettman, & Johnson, 1988; Gigerenzer & Selten, 2002). Yet, being a skilled carpenter requires more than a toolbox: you also have to know when to use which tool. Todd and Gigerenzer (2012) postulate that we choose heuristics that are well-adapted to our current situation (i.e. *ecologically rational*), but they do not explain how we are able to do so. Empirical evidence suggests that people do indeed choose heuristics adaptively (Payne et al., 1988; Bröder, 2003; Pachur, Todd, Gigerenzer, Schooler, & Goldstein, 2011). Despite some progress (Shrager & Siegler, 1998; Erev & Barron, 2005; Rieskamp & Otto, 2006), the problem of formulating the computational principles of strategy selection remains unsolved (Marewski & Link, 2014).

According to previous theories of strategy selection we learn to choose the strategy that works best *on average* across all problems in an environment (Rieskamp & Otto, 2006; Erev & Barron, 2005) or category (Shrager & Siegler, 1998). This approach ignores that every problem has distinct characteristics that determine the strategies' differential effectiveness. These *context-free* accounts therefore predict the formation of rigid mental habits that always pursue the same strategy when the environment is stable, whereas people can flexibly switch between cognitive strategies (Payne et al., 1988; Lieder et al., 2014). By contrast, our theory posits that people learn a mental model predicting the effectiveness of cognitive strategies from features of the problem to be solved. We evaluate our model against empirical data and two previous theories of strategy selection: SSL (Rieskamp & Otto, 2006) and RELACS (Erev & Barron, 2005) which use model-free reinforcement learning to estimate how well each strategy performs *on average* across all problems.

2 Model-based strategy selection learning

Strategy selection is a metacognitive decision with uncertain consequences. We therefore leveraged *rational metareasoning* – a decision-theoretic framework for choosing computations (Russell & Wefald, 1991) – to develop a rational model of strategy selection that is theoretically sound, computationally efficient, and competitive with state-of-the-art algorithm selection methods (Lieder et al., 2014). Rational metareasoning chooses the strategy $s^\star$ with the highest value of computation (VOC) for the problem specified by input $\mathbf{i}$:

$$s^\star = \arg\max_{s \in \mathcal{S}} \mathrm{VOC}(s, \mathbf{i}), \tag{1}$$

where $\mathcal{S}$ is the set of the agent's cognitive strategies $\mathcal{S}$. The VOC of executing a cognitive strategy s is the expected net increase in utility over acting without deliberation. If the strategy chooses an action and the utility of the available actions remains approximately constant while the agent deliberates, then the VOC can be approximated by the increase in the expected return due the resulting action minus the opportunity cost of the strategy's execution time T:

$$\mathrm{VOC}(s; \mathbf{i}) \approx \mathbb{E}\left[R | s, \mathbf{i}\right] - \mathbb{E}\left[\mathrm{TC}(T) | s, \mathbf{i}\right], \tag{2}$$

where R is the increase in reward and $\mathrm{TC}(T)$ is the opportunity cost of running the algorithm for T units of time. The reward R can be binary (correct vs. incorrect output) or numeric (e.g., the payoff). Equations 1-2 reveal that near-optimal strategy selection can be achieved by learning to predict the strategies' expected rewards and execution times from features $\mathbf{f}(\mathbf{i})$ of the input $\mathbf{i}$ that specifies the problem to be solved. These predictions can be learned by Bayesian linear or logistic regression as described in Lieder et al. (2014).

Equation 1 is optimal when the VOC is known, but when the VOC is unknown the value of exploration should not be ignored. To remedy this problem, we employ Thompson sampling (Thompson, 1933)–a near optimal solution to the exploration-exploitation dilemma (May, Korda, Lee, & Leslie, 2012). Concretely, each heuristic s is chosen ($S = s$) according to the probability that its VOC is maximal:

$$P(S = s) \propto P\left(s = \arg\max_s \mathrm{VOC}(s; \mathbf{i})\right). \tag{3}$$

For more information on our model and methodology please consult Lieder and Griffiths (2015).

3 Learning when to use fast-and-frugal heuristics

Fast-and-frugal heuristics perform very few computations and use only a small subset of the available information (Gigerenzer & Selten, 2002). For instance, the Take-the-Best heuristic for multi-attribute decision-making chooses the option with the highest value on the most predictive attribute that distinguishes the options and ignores all other attributes. This strategy works in so-called *non-compensatory* environments in which the attributes' predictive validities fall off so rapidly that the decision recommended by the most predictive attribute cannot be overturned by rationally incorporating some or all of the remaining attributes. Yet it can fail miserably in compensatory environments in which

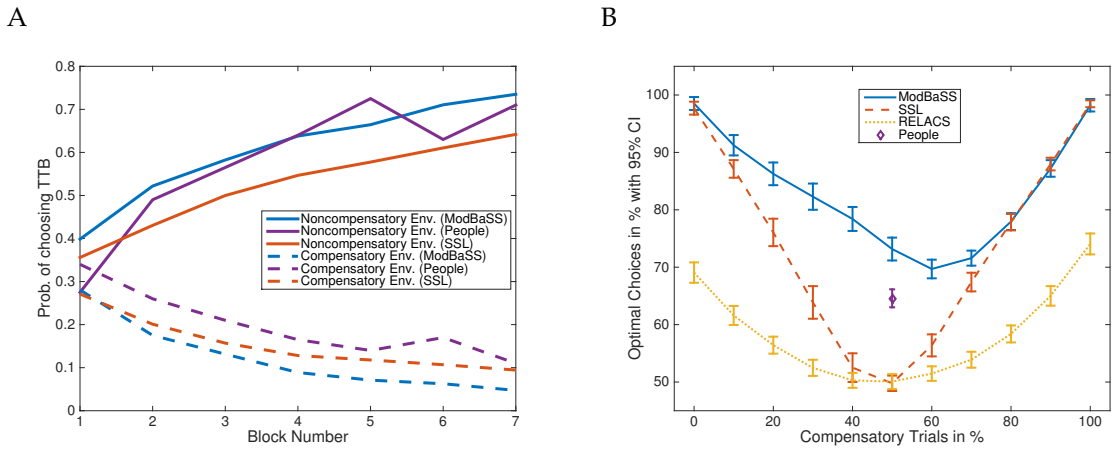


Figure 1: A: Model-based strategy selection (ModBaSS) explains findings by Rieskamp and Otto (2006).
B: Model-based strategy selection (ModBaSS) outperforms SSL and RELACS–especially in heterogeneous environments.

no single attribute reliably identifies the best choice by itself. In those environments the Weighted-Additive strategy (WADD), which computes a weighted average of all attributes (e.g. a gamble's expected value), is often more effective.

Bröder (2003) found that people use Take-the-Best more frequently in non-compensatory environments than in compensatory environments. Rieskamp and Otto (2006) conducted an experiment suggesting that this adaptation results from reinforcement learning. As a first test of our model we demonstrate that it can explain the findings reported by Rieskamp and Otto (2006). Rieskamp and Otto's first experiment was structured into seven blocks comprising 24 trials each. In each trial participants chose between two investment options based on five binary attributes whose predictive validities were constant and explicitly stated. To apply our general model-based strategy selection theory to multi-attribute decision making, we chose the following features $\mathbf{f} = (f_1, f_2, f_3)$: the validity of the most reliable discriminative cue (f_1), the gap between the validity of the most reliable cue favoring the first option and the most reliable cue favoring the second option (f_2), and the absolute difference between the number of attributes favoring the first option and the second option respectively (f_3). The simulated agent's toolbox contained two strategies: Take-the-Best and the Weighted-Additive strategy. We created compensatory and non-compensatory environments similar to those used by Rieskamp and Otto (2006): In the non-compensatory environment Take-the-Best always makes the Bayes-optimal decision, and in the compensatory environment the Weighted-Additive strategy always makes the Bayes-optimal decision. In both environments Take-the-Best and the Weighted-Additive strategy make the same decision on exactly half of the trials.

We found that model-based strategy selection can explain people's ability to adapt to compensatory as well as non-compensatory environments (see Figure 1A): When the environment was non-compensatory our model learned to use Take-the-Best. Conversely, when the environment was non-compensatory our model learned to avoid Take-the-Best and use the Weighted-Additive strategy instead. Our simulation results show that model-based strategy selection captured that participants gradually adapted their strategy choices to the decision environment. However, SSL explains people's choices equally well; the mean squared errors of the fits achieved by model-based strategy selection and SSL were almost identical (0.0050 vs. 0.0048); see Figure 1A.

4 Strategy selection in mixed environments

Since model-based and context-free strategy selection can both explain the results of Rieskamp and Otto (2006), a new experiment is needed to test if people learn a model enabling them to predict the strategies' effectiveness from features of the problem to be solved. We thus investigated under which conditions model-based strategy choices differ from those of SSL and RELACS. Concretely, we evaluated the performance of context-free versus model-based strategy selection in 11 environments with $p \in \{0\%, 10\%, 20\%, \cdots, 100\%\}$ compensatory and $1 - p$ non-compensatory problems. Our simulations revealed that the performance of context-free strategy selection drops rapidly with the variance in the environment's compensatoriness whereas model-based strategy selection is much less susceptible to it (see Figure 1B): As the ratio of compensatory to non-compensatory problems approaches $50/50$ the performance of SSL and RELACS drops to the chance level. By contrast, the performance of model-based strategy selection, remains above 70%. The reason for this difference is that model-based strategy selection learns to use Take-the-Best for non-compensatory problems and the Weighted-Additive strategy for compensatory problems whereas SSL and RELACS learn to always use the same strategy. We can therefore determine whether people use context-free or model-based strategy selection by measuring their performance in a heterogeneous environment with the following experiment:

4.1 Methods

We recruited 120 participants on Amazon Mechanical Turk. Each participant was paid 50 cents for about five minutes of work. The experiment comprised 30 binary decisions. Participants played the role of a banker deciding which of two companies receives a loan based on the companies' ratings on six criteria. The decision problems were chosen such that Take-the-Best and the Weighted-Additive strategy make opposite decisions on every trial. In half of the trials, the decision of Take-the-Best was correct and in half of the trials the decision of the Weighted-Additive strategy was correct. Thus, always using Take-the-Best, always using the Weighted-Additive strategy, choosing one of the two strategies at random, or context-free strategy selection would result in an accuracy of 50%; see Figure 1B.

4.2 Results and Discussion

People chose the more creditworthy company in 64.6% of the trials (99% CI: $[62.5\%; 66.6\%]$). We can thus conclude that people performed significantly better than chance ($p < 10^{-15}$). This is qualitatively consistent with model-based strategy selection but inconsistent with context-free strategy selection; see Figure 1B.

In conclusion, people's performance in homogeneous decision environments is consistent with model-based and context-free strategy selection, but context-free strategy selection is insufficient to explain human performance in heterogeneous environments whereas model-based strategy selection can account for it. Thus our results suggest people use model-based strategy selection.

5 Adaptive flexibility in strategy selection

People adapt their strategy not only to reoccurring situations, but they can also flexibly switch strategies as soon as the situation changes. This flexibility has been empirically demonstrated in decision-making under risk: Payne et al. (1988) found that people adaptively switch decision strategies in the absence of feedback.

To determine if model-based strategy selection can explain this finding we simulated Experiment 1 from Payne et al. (1988). This experiment comprised ten instances of each of four types of decision problems that were presented in random order. The four problem types were defined by the time constraint (15 seconds vs. none) and the dispersion of the outcomes' probabilities (low vs. high). In each problem participants chose between four gambles. The four gambles assigned different payoffs to four possible outcomes but they shared the same outcome probabilities. Payne et al. (1988) measured the use of fast-and-frugal attribute-based heuristics, namely Take-the-Best or Elimination-by-Aspects (EBA; Tversky, 1972), by the proportion of time their participants spend processing the options' payoffs for the most probable outcome. For the compensatory Weighted-Additive strategy this proportion is only 25%, but for Take-the-Best and Elimination-by-Aspects it is up to 100%. When the dispersion of outcome probabilities was high, people focused more on the most probable outcome, and time pressure also increased people's propensity for selective and attribute-based processing; see Figure 2. Thus, people seem to use non-compensatory strategies such as Take-the-Best and Elimination-by-Aspects more frequently when time is limited or some outcomes are much more probable than others.

We performed 1000 simulations of people's strategy choices in this experiment. We simulated people's prior learning experiences about risky choice strategies by applying model-based strategy selection learning to ten instances of each of the 144 types of decision problems considered by Payne et al. (1988). We then applied model-based strategy selection with the learned model of the strategies' performance to a simulation of Experiment 1 from Payne et al. (1988). Since their participants received no feedback, our simulation assumed no learning during the experiment. Model-based strategy selection correctly predicted that time-pressure and probability dispersion increase people's propensity to use Take-the-Best or Elimination-by-Aspects; see Figure 2. SSL and RELACS, by contrast, predict that there should be no difference between the four conditions. This is because SSL and RELACS cannot learn to choose different strategies for different kinds of problems. Their strategy choices only change in response to reward or punishment but the experiment provided neither. In conclusion, model-based strategy selection can account for adaptively flexible strategy-selection in decision-making under risk but SSL and RELACS cannot.

6 General Discussion

We have proposed a rational solution to the strategy selection problem: model-based strategy selection learning. The primary difference to previous accounts is that our model chooses strategies based on the distinct characteristics of individual problems, whereas SSL and RELACS learn which strategy works best on average across all problems. Thus, people's adaptive flexibility appears to require learning to predict the strategies' effectiveness from features of the problem to be solved. We have previously found that model-based strategy selection can account for people's adaptive strategy selection in sorting numbers but previous theories cannot (Lieder et al., 2014). Here we have shown that this conclusion also holds for our capacity to select decision-strategies.

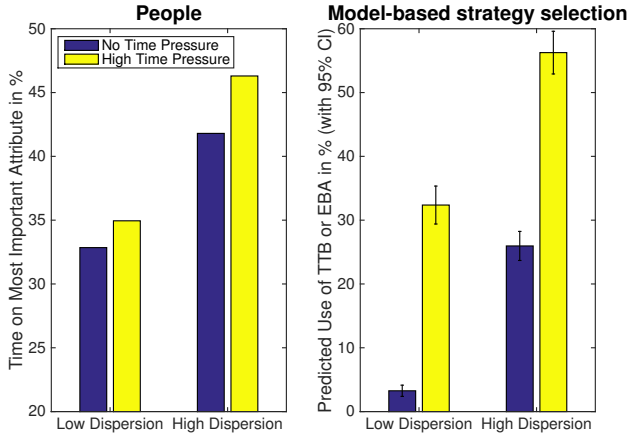


Figure 2: Model-based strategy selection predicts the increase in selective attribute-based processing with dispersion and time pressure observed by Payne et al. (1988).

In conclusion, model-based strategy selection is a promising framework for cognitive modeling. It could, for instance, be used to explain paradoxical inconsistencies in risky choice by identifying why people use different heuristics in different contexts. Furthermore, our model of strategy selection learning can be applied to education and cognitive training: First, our model could be used to optimize problem sets for helping students learn when to apply which procedure (e.g. in algebra) rather than drilling them on one procedure at a time. Second, our model could be used to design cognitive training programs promoting adaptive flexibility in decision making and beyond. Future work will also explore learning the VOC of elementary information processing operations (Russell & Wefald, 1991) as a model of strategy discovery.

References

Bröder, A. (2003). Decision making with the" adaptive toolbox": Influence of environmental structure, intelligence, and working memory load. *J. Exp. Psychol.-Learn. Mem. Cogn.*, *29*(4), 611.

Erev, I., & Barron, G. (2005). On adaptation, maximization, and reinforcement learning among cognitive strategies. *Psychological review*, *112*(4), 912–931.

Gigerenzer, G., & Selten, R. (2002). *Bounded rationality: The adaptive toolbox*. MIT Press.

Lieder, F., & Griffiths, T. L. (2015). When to use which heuristic: A rational solution to the strategy selection problem. In D. C. Noelle & R. Dale (Eds.), *Proceedings of the 37th annual conference of the cognitive science society.* Austin, TX: Cognitive Science Society.

Lieder, F., Plunkett, D., Hamrick, J. B., Russell, S. J., Hay, N., & Griffiths, T. (2014). Algorithm selection by rational metareasoning as a model of human strategy selection. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Weinberger (Eds.), *Adv. neural inf. process. syst. 27* (pp. 2870–2878). Curran Associates, Inc.

Marewski, J. N., & Link, D. (2014). Strategy selection: An introduction to the modeling challenge. *Wiley Interdisciplinary Reviews: Cognitive Science*, *5*(1), 39–59.

May, B. C., Korda, N., Lee, A., & Leslie, D. S. (2012). Optimistic Bayesian sampling in contextual-bandit problems. *J. Mach. Learn. Res.*, *13*.

Pachur, T., Todd, P. M., Gigerenzer, G., Schooler, L. J., & Goldstein, D. G. (2011). The recognition heuristic: a review of theory and tests. *Frontiers in psychology*, *2*.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1988). Adaptive strategy selection in decision making. *J. Exp. Psychol.-Learn. Mem. Cogn.*, *14*(3), 534.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1993). *The adaptive decision maker*. Cambridge University Press.

Rieskamp, J., & Otto, P. E. (2006). SSL: A theory of how people learn to select strategies. *J. Exp. Psychol. Gen.*, *135*(2), 207–236.

Russell, S., & Wefald, E. (1991). Principles of metareasoning. *Artificial Intelligence*, *49*(1-3), 361–395.

Shrager, J., & Siegler, R. S. (1998). SCADS: A model of children's strategy choices and strategy discoveries. *Psychological Science*, *9*(5), 405–410.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 285–294.

Todd, P. M., & Gigerenzer, G. (2012). *Ecological rationality: Intelligence in the world*. Oxford University Press.

Tversky, A. (1972). Elimination by aspects: A theory of choice. *Psychological review*, *79*(4), 281.

A Biologically Plausible 3-factor Learning Rule for Expectation Maximization in Reinforcement Learning and Decision Making

Mohammad Javad Faraji
School of life sciences, Brain mind institute
School of computer and communication sciences
École Polytechnique Fédéral de Lausanne (EPFL)
Lausanne, CH-1015
mohammadjavad.faraji@epfl.ch

Kerstin Preuschoff
Geneva finance research institute
School of economics and management
University of Geneva
Geneva, CH-1211
kerstin.preuschoff@unige.ch

Wulfram Gerstner
School of life sciences, Brain mind institute
School of computer and communication sciences
École Polytechnique Fédéral de Lausanne (EPFL)
Lausanne, CH-1015
wulfram.gerstner@epfl.ch

Abstract

One of the most frequent problems in both decision making and reinforcement learning (RL) is expectation maximization involving functionals such as reward or utility. Generally, these problems consist of computing the optimal solution of a density function. Instead of trying to find this exact solution, a common approach is to approximate it through a learning process.

In this work we propose a functional gradient rule for the maximization of a general form of density-dependent functionals using a stochastic gradient ascent algorithm. If a neural network is used for parametrization of the desired density function, the proposed learning rule can be viewed as a modulated Hebbian rule. Such a learning rule is biologically plausible, because it consists of both local and global factors corresponding to the coactivity of pre/post-synaptic neurons and the effect of neuromodulation, respectively.

We first apply our technique to standard reward maximization in RL. As expected, this yields the standard policy gradient rule in which parameters of the model are updated proportional to the amount of reward. Next, we use variational free energy as a functional and find that the estimated change in parameters is modulated by a measure of surprise signal. Finally, we propose an information theoretical equivalent of existing models in expected utility maximization, as a standard model of decision making, to incorporate both individual preferences and choice variability. We show that our technique can also be applied into such novel framework.

Keywords: Stochastic gradient ascent, reinforcement learning, decision making, expected utility maximization, multi-factor learning rules, free energy, surprise, neuromodulation.

Acknowledgements

This project is funded by the European Research Council (grant agreement no. 268 689).

1 Introduction

Theoretical descriptions of synaptic plasticity have been dominated by Hebb's rule [1] which is based on two major factors: *locality* and *coactivity*. According to Hebb's rule, both pre- and post-synaptic neurons have to be active to make their connection stronger. Empirical studies, however, show the existence of other *global factors* that can influence synaptic plasticity [2]. These global factors correspond to diffusive action of neuromodulators or feedback from the activity state of a whole population. Deficits in activity of the neuromodulatory system (corresponding to global factors) in humans and animals leaves many tasks un-learnable [3]. For instance, Dopamine (DA) as a neuromodulator is used in signaling reward prediction error that takes part in temporal difference (TD) learning algorithms such as Q-learning and SARSA [4]. Acetylcholine (Ach) is another candidate neuromodulator used in signaling alertness [5]. It is thus of interest to expand on Hebbian learning rules and formulate general new synaptic plasticity rules that combine two Hebbian activity factors with one or multiple global factors. The simplest 3-factor learning rule, including two Hebbian terms modulated by a third factor, is an example.

Expectation maximization, on the other hand, is one of the most frequently encountered problems in both decision making [6] and reinforcement learning (RL) [4]. It consists of computing the optimal solution of a density function. It might represent a learning agent's policy in RL, or the likelihood of selecting different choices in a decision making process. We introduce a functional gradient rule for the maximization of a general form of density-dependent functionals, such as reward or utility, using a stochastic gradient ascent algorithm. We obtain a learning rule by which we approximate the optimal solution through a learning process. This learning rule benefits from a biological plausibility if a neural network is used for parametrization of the desired density function. This is consistent with a modulated Hebbian learning rule (i.e., 3-factor learning rule) in which both global and local factors influence the synaptic connections among the neurons. We apply our technique to standard reward maximization in RL and a variational learning problem to show that reward and surprise signals can be interpreted as third factors in this framework. We further propose a more general formalism of expected utility maximization, a standard model of decision making, that can be solved using functional gradient rule. The aim of such a novel approach is to incorporate both individual preferences and choice variability in the decision making process regardless of the specific details of the model used.

2 Methods

We apply a stochastic gradient ascent technique to approximate the optimal density function that maximizes a functional $\mathbb{F}[P] = \langle \mathcal{F}[P] \rangle_P$ where $\langle . \rangle_P$ denotes the average with respect to the probability density $P(x)$ of the random variable X. The term $\mathcal{F}[P]$ might be considered as a general form of reward or utility function which itself depends on the density function P. The general form of the online gradient rule will be derived in the following.

Theorem 1 (functional gradient rule): The stochastic gradient ascent algorithm for maximizing a functional $\mathbb{F}[P] = \langle \mathcal{F}[P] \rangle_P$ over all possible distributions P parametrized by $\theta \in \mathbb{R}^n$ yields the online learning rule,

$$\Delta\theta \propto \tilde{\mathcal{F}} \nabla_\theta \ln P, \tag{1}$$

where the multiplier factor $\tilde{\mathcal{F}}$ is defined as

$$\tilde{\mathcal{F}} = \frac{\partial}{\partial P} \left(P \mathcal{F}[P] \right) = \mathcal{F}[P] + \frac{\partial \mathcal{F}[P]}{\partial \ln P}. \tag{2}$$

Proof: In order to have an online learning rule for $\theta \in \mathbb{R}^n$, we need to find a term $\Delta\theta$ such that $\langle \Delta\theta \rangle_P = \nabla_\theta \mathbb{F}[P]$. Using the nice trick $P \nabla_\theta \ln P = \nabla_\theta P$ we have

$$\begin{aligned}
\nabla_\theta \langle \mathcal{F}[P] \rangle_P &= \int dx\ P \nabla_\theta \mathcal{F}[P] + \mathcal{F}[P] \nabla_\theta P = \int dx\ P \left(\frac{\partial \mathcal{F}[P]}{\partial P} \nabla_\theta P \right) + \mathcal{F}[P] \left(P \nabla_\theta \ln P \right) \\
&= \left\langle \frac{\partial \mathcal{F}[P]}{\partial P} \nabla_\theta P + \mathcal{F}[P] \nabla_\theta \ln P \right\rangle_P = \left\langle \frac{\partial \mathcal{F}[P]}{\partial P} \left(P \nabla_\theta \ln P \right) + \mathcal{F}[P] \nabla_\theta \ln P \right\rangle_P \\
&= \left\langle \left(\frac{\partial \mathcal{F}[P]}{\partial P} P + \mathcal{F}[P] \right) \nabla_\theta \ln P \right\rangle_P = \left\langle \frac{\partial \left(P \mathcal{F}[P] \right)}{\partial P} \nabla_\theta \ln P \right\rangle_P .
\end{aligned} \tag{3}$$

Corollary 1: The multiplier factor $\tilde{\mathcal{F}}$ in the learning rule (1) can be replaced by $\tilde{\mathcal{F}} + c$ where $c \in \mathbb{R}$ is a constant term because

$$\left\langle \left(\tilde{\mathcal{F}} + c \right) \nabla_\theta \ln P \right\rangle_P = \left\langle \tilde{\mathcal{F}} \nabla_\theta \ln P \right\rangle_P + c \left\langle \nabla_\theta \ln P \right\rangle_P , \tag{4}$$

and $\langle \nabla_\theta \ln P \rangle_P = \int dx\ P \nabla_\theta \ln P = \int dx\ \nabla_\theta P = \nabla_\theta \int dx\ P = \nabla_\theta (1) = 0$.

Corollary 2: If $\mathcal{F}[P]$ does not explicitly depend on P or is linear in $\ln P$, then the multiplier factor $\tilde{\mathcal{F}}$ can be replaced by $\mathcal{F}[P]$. The proof is simply done by using **Corollary 1** in (2).

We want to stress that our proposed learning rule (1) can indeed be embedded in the class of biologically plausible 3-factor learning rules, if a neural network is used for parametrization. The term $\tilde{\mathcal{F}}$ represents a globally modulating third factor, depending on the properties of the neuronal ensemble in a nonlocal fashion. The term $\nabla_\theta \ln P$ represents a Hebbian term, which can be shown to depend on both pre- and postsynpatic activity. As an example, we use a population of spiking neural network for learning the density function P. The neuron model that we use here is a generalized linear model (GLM). This model has the form of a Spike Response Model (SRM) with escape noise [7, 8]. The membrane potential $u_i(t)$ of neuron i at time t is given as $u_i(t) = \sum_j w_{ij}(X_j * \phi)(t) + \eta_i(t)$, where w_{ij} is the synaptic efficacy between pre-synaptic neuron j and post-synaptic neuron i, $X_j(t) = \sum_f \delta(t - t_j^f)$ denotes the presynaptic spike train, $\phi(t)$ is the somatic EPSP, and $\eta_i(t) = -\eta_0 \int_0^t ds\ e^{-\frac{t-s}{\tau_a}}\ X_i(s)$ is the adaptation potential (η_0 and τ_a are constants). The spikes are then generated by a stochastic Poisson process with an exponential escape rate $\rho_i(t)$ [8] conditioned on the membrane potentials,

$$\rho_i(t) = \rho_0 \exp(\frac{u_i(t) - \theta}{\Delta U}), \tag{5}$$

where θ and ΔU are physical constants of the neuron. Free parameters $\theta \in \mathbb{R}^n$ by which P is parametrized are synaptic efficacies w_{ij} between neurons. Each sampled observed data x is modeled as a set of spike trains $\{X_i\}$ generated by all the neurons within a neuronal population. $P(x)$ is then modeled as the likelihood of generating each set of spike trains, corresponding to each sampled observed data x, in that population. The likelihood of a particular spike train $x = \{X_i\}$ which is observed in the interval $[0, T]$ can be written as [9, 10]

$$\ln P(x) = \sum_k \int_0^T dt\ [\ln \rho_k(t) X_k(t) - \rho_k(t)], \tag{6}$$

and its gradient with respect to the particular synaptic weight w_{ij} is calculated as (see [10, 11] for details)

$$\nabla_{w_{ij}} \ln P(x) = \frac{1}{\Delta U}\ (X_j * \phi)(t)\ [X_i(t) - \rho_i(t)]. \tag{7}$$

Therefore, we conclude that the learning rule for synaptic weights w_{ij} according to gradient ascent $\Delta w_{ij} \propto \nabla_{w_{ij}} \ln P(x)$ can be calculated locally and is written as a product of two local (Hebbian) factors: $(X_j * \phi)(t)$ which depends on the pre-synaptic neuron j and $[X_i(t) - \rho_i(t)]$ that depends on the state of the post-synaptic neuron i.

3 Results

In this section we describe two examples of using our proposed functional gradient rule. First, reward maximization in the context of RL can be formulated as finding the optimal policy $\pi(a|s)$ that maximizes the expected reward $\langle R(s,a) \rangle_{\pi(a|s)f(s)}$ where $R(s,a)$ denotes the reward for taking action a in state s and $f(s)$ is the density function of state space. The online learning rule (1) for reward maximization in RL is

$$\Delta\theta \propto R \nabla_\theta \ln \pi, \tag{8}$$

where θ is used to parametrize policy π. Note that since reward $R(s,a)$ does not explicitly depend on the policy π, the multiplier factor $\tilde{\mathcal{F}}$ in (1) is the reward $R := R(s,a)$ itself, according to **Corollary 2**. The learning rule (8) is the standard policy gradient rule used in the reward maximization approach known as R-max [12].

Second, variational methods are typically used in complex statistical models which are defined by a joint distribution $p(v,h)$ over a set of observed (visible) v and unobserved (hidden) h variables. The joint distribution p is known as a generative model governed by some adaptive parameters $\theta \in \mathbb{R}^n$. Two main purposes of using variational methods are to analytically approximate the posterior distribution $p(h|v)$ of hidden variables (for statistical inference over them) or to derive a lower bound for a marginal likelihood $p(v) = \sum_h p(v,h)$ of the visible variables (usually for model selection). A computationally tractable lower bound $\mathcal{L}(q; w, \theta)$ for the marginal likelihood $p(v)$ of the visible variables is calculated as

$$\begin{aligned} \ln p(v) &= \ln \sum_h p(v,h) = \ln \sum_h q(h|v)\ \frac{p(v,h)}{q(h|v)} \\ &\geq \sum_h q(h|v)\ \ln\ \frac{p(v,h)}{q(h|v)} := \mathcal{L}(q; w, \theta), \end{aligned} \tag{9}$$

where we have applied Jensen's inequality. Here $w \in \mathbb{R}^m$ denotes adaptive parameters used for expressing $q(h|v)$. It is easy to see that the difference between the true log likelihood $\ln p(v)$ and its approximated lower bound $\mathcal{L}(q; w, \theta)$ is

$$\ln p(v) - \mathcal{L}(q; w, \theta) = \sum_h q(h|v)\ \frac{q(h|v)}{p(h|v)} := D_{KL}(q||p). \tag{10}$$

Therefore, maximizing the lower bound $\mathcal{L}(q;w,\theta)$ is equivalent to minimizing the Kullback-Leibler divergence $D_{KL}(q||p)$ of the true posterior distribution $p(h|v)$ from the approximated one $q(h|v)$. The lower bound $\mathcal{L}(q;w,\theta)$ is known as (negative) variational free energy $\mathbb{F}[q;v]$ in statistical learning [13] that can be expressed as

$$\mathbb{F}[q;v] = -\mathcal{L}(q;w,\theta) = \langle -\ln p(v,h)\rangle_q - H(q). \tag{11}$$

The variational free energy $\mathbb{F}[q;v]$ for each observed variable v can be considered as a measure of its novelty indicating how much the new observed data v is surprising. Here surprise is taken to be the negative log-likelihood $-\ln p(v)$ of observed data v. One can express variational free energy $\mathbb{F}[q;v]$ as $\langle \mathcal{F}[q;v]\rangle_q$ where $\mathcal{F}[q;v] = -\ln p(v,h) + \ln q(h|v)$, denotes the *instantaneous* amount of free energy for observed data v. The online learning rule, suggested by **Theorem 1**, for variational free energy minimization is then given by

$$\Delta w \propto -\mathcal{F}\nabla_w \ln q, \tag{12}$$

where the minus sign is because of the minimization and $\mathcal{F} = \mathcal{F}[q;v]$ is the instantaneous amount of free energy for observed data v. Note that since $\mathcal{F}[q;v]$ is linear in $\ln q$, the multiplier factor $\tilde{\mathcal{F}}$ in (1) would be equal to $\mathcal{F}[q;v]$ according to **Corollary 2**. The significance of learning rule (12) is that it explicitly shows that the amount of estimated change in parameters w for learning q is proportional to the amount of surprise or information contained in the observed data v. In other words, the surprise signal measured by the instantaneous free energy modulates the learning rate such that surprising observed data v yields more change of the parameter w for learning the approximate posterior distribution q.

4 Discussion

A standard model of decision making is *expected utility maximization* [14] in which a decision maker selects a choice $x^* \in \mathcal{X}$ with the highest *subjective expected utility* $U(x^*)$ among all other alternatives $x \in \mathcal{X}$. In a probabilistic framework, it can be interpreted as selecting choice x^* with probability 1 and choosing the rest with probability 0 (i.e. $P(x) = \delta(x - x^*)$ is the corresponding choice selection density function which determines the likelihood of selecting different choices, where $\delta(.)$ denotes the Kronecker delta function). The density function $P(x) = \delta(x - x^*)$ maximizes the expected value of the utility function $\langle U(x)\rangle_P$ among all possible density functions $P(x)$ because

$$\langle U(x)\rangle_P = \sum_x U(x)P(x) \le \sum_x U(x^*)P(x) = U(x^*) = \langle U(x)\rangle_{\delta(x-x^*)}. \tag{13}$$

In reinforcement learning, however, choosing the action with the highest value function does not allow for sufficient exploration; this requires choice variability, e.g., by adding noise. Furthermore, individual preferences should be incorporated into the decision making processes, such as action selection in RL.

Expected utility theory accounts for individual differences by explicitly modeling different beliefs about the probabilities of different outcomes. Instead of using a stochastic action selection function (such as a sigmoid) we propose an information theoretical equivalent of existing models to incorporate both individual preferences and choice variability. As such we do not need to impose any specific form of constraints existing in different models. In contrast to maximizing just the average utility $\langle U(x)\rangle_P$, maximizing the functional

$$\begin{aligned}\mathbb{F}[P] &= \langle U(x)\rangle_P + \frac{1}{\lambda_1}H(P) - \frac{1}{\lambda_2}H(P,P_0)\\ &= \left\langle U(x) - \frac{1}{\lambda_1}\ln P(x) + \frac{1}{\lambda_2}\ln P_0(x)\right\rangle_P,\end{aligned} \tag{14}$$

yields a choice selection density function $P(x)$ which not only leads to a relatively high average utility, but also allows exploration. Further, it does not allow the solution to be highly different from a reference P_0. While the entropy $H(P) = \langle -\ln P(x)\rangle_P$ of a density function P in (14) models choice variability, the relative entropy $H(P,P_0) = \langle -\ln P_0(x)\rangle_P$ models subjectivity by involving a subjective reference density function P_0. The minus sign in the last term of the first line in (14) penalizes those density functions that are highly different from a subjective reference P_0. The parameters λ_1 and λ_2 control the fuzziness of the solution by changing the weights of the second and third terms, respectively. By taking the derivative of (14) with respect to P and setting it equal to zero, one could find that the functional (14) is maximized by

$$P^*(x) = \arg\max_P \mathbb{F}[P] = \frac{P_0(x)^{\frac{\lambda_1}{\lambda_2}}\, e^{\lambda_1 U(x)}}{Z(\lambda_1,\lambda_2)}, \tag{15}$$

where $Z(\lambda_1,\lambda_2) = \sum_x P_0(x)^{\frac{\lambda_1}{\lambda_2}}\, e^{\lambda_1 U(x)}$ is the normalizing factor. Equation (15) resembles a (modified) Bayes' rule in the sense that the effect of utility U in making the posterior density function P^* is controlled by a free parameter λ_1 and a prior belief P_0 that is affected by the ratio $\frac{\lambda_1}{\lambda_2}$. Although the optimal density P^* that yields the maximal functional value

$\mathbb{F}[P^*] = \frac{1}{\lambda_1}\ln Z(\lambda_1,\lambda_2)$ is explicitly derived in (15), it can also be learned using the functional gradient rule (1). This is because the functional (14) can be expressed as $\langle \mathcal{F}[P] \rangle_P$ where $\mathcal{F}[P] = U(x) - \frac{1}{\lambda_1}\ln P(x) + \frac{1}{\lambda_2}\ln P_0(x)$ is a density-dependent functional.

If the maximizer P^* is approximated by any other density $\tilde{P}$, then its corresponding functional value $\mathbb{F}[\tilde{P}]$ differs from its maximal value $\mathbb{F}[P^*]$ in proportion to the KL divergence $D_{KL}(\tilde{P}||P^*) \geq 0$. This is because,

$$\begin{aligned}
\mathbb{F}[P^*] - \mathbb{F}[\tilde{P}] &= \frac{1}{\lambda_1}\ln Z(\lambda_1,\lambda_2) - \left\langle U(x) - \frac{1}{\lambda_1}\ln \tilde{P}(x) + \frac{1}{\lambda_2}\ln P_0(x) \right\rangle_{\tilde{P}} \\
&= \frac{1}{\lambda_1}\left\langle \ln Z(\lambda_1,\lambda_2) - \ln e^{\lambda_1 U(x)} + \ln \tilde{P}(x) - \frac{\lambda_1}{\lambda_2}\ln P_0(x) \right\rangle_{\tilde{P}} \\
&= \frac{1}{\lambda_1}\left\langle \ln \frac{Z(\lambda_1,\lambda_2)\tilde{P}(x)}{e^{\lambda_1 U(x)} P_0(x)^{\frac{\lambda_1}{\lambda_2}}} \right\rangle_{\tilde{P}} = \frac{1}{\lambda_1}\left\langle \ln \frac{\tilde{P}(x)}{P^*(x)} \right\rangle_{\tilde{P}} = \frac{1}{\lambda_1} D_{KL}\left(\tilde{P}||P^*\right). \qquad (16)
\end{aligned}$$

As an example, we investigate a binary decision making task (such as the two-armed bandit problem) in which a subject has to make a decision between two alternatives $x = 1$ and $x = 0$. The probability $P(x)$ of making decision x is modeled by a Bernoulli distribution parametrized by θ such that $P(x) = \theta^x(1-\theta)^{(1-x)}$. We use $P_0(x) = 0.5$ to incorporate no *a priori* preference in making different decisions. We further assume that $\lambda_1 = \lambda_2 = \lambda$ to make the formula simpler. As such, the optimal probability of making the decision $x = 1$ in our binary example is equal to

$$P^*(x=1) = \frac{e^{\lambda U(x=1)}}{e^{\lambda U(x=1)} + e^{\lambda U(x=0)}} = \frac{1}{1 + e^{-\lambda \Delta U}}, \qquad (17)$$

where $\Delta U = U(x=1) - U(x=0)$ is the difference between the decisions' utilities. If $U(x=1) > U(x=0)$, the probability $P^*(x=1)$ of making decision $x = 1$ in (17) is greater than 0.5. The parameter λ then determines how big that probability should be for different values of ΔU. Note that the stochastic (sigmoid) action selection function, which is used in the expected utility theorem for modeling choice variability, is explicitly derived in (17) as the optimal solution in the sense that it maximizes the functional (14).

References

[1] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2002.

[2] John NJ Reynolds and Jeffery R Wickens. Dopamine-dependent plasticity of corticostriatal synapses. *Neural Networks*, 15(4):507–521, 2002.

[3] Michael W Decker and James L McGaugh. The role of interactions between the cholinergic system and other neuromodulatory systems in learing and memory. *Synapse*, 7(2):151–168, 1991.

[4] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998.

[5] Michael I Posner and Jin Fan. Attention as an organ system. *Topics in integrative neuroscience: From cells to cognition*, pages 31–61, 2004.

[6] Irving L Janis and Leon Mann. *Decision making: A psychological analysis of conflict, choice, and commitment*. Free Press, 1977.

[7] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

[8] Renaud Jolivet, Alexander Rauch, Hans-Rudolf Lüscher, and Wulfram Gerstner. Predicting spike timing of neocortical pyramidal neurons by simple threshold models. *Journal of computational neuroscience*, 21(1):35–49, 2006.

[9] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural computation*, 18(6):1318–1348, 2006.

[10] Danilo J Rezende, Daan Wierstra, and Wulfram Gerstner. Variational learning for recurrent spiking networks. In *NIPS*, pages 136–144, 2011.

[11] Danilo Jimenez Rezende and Wulfram Gerstner. Stochastic variational learning in recurrent spiking networks. *Frontiers in computational neuroscience*, 8, 2014.

[12] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[13] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[14] Jack Meyer. Two-moment decision models and expected utility maximization. *The American Economic Review*, pages 421–430, 1987.

The formation of habits: a computational model mixing reinforcement and Hebbian learning

Meropi Topalidou
INRIA Bordeaux Sud-Ouest, Bordeaux, France
Université de Bordeaux, CNRS UMR 5293, IMN, France
LaBRI, Université de Bordeaux, IPB, CNRS, UMR 5800, Talence, France
meropi.topalidou@inria.fr

Daisuke Kase
Université de Bordeaux, CNRS UMR 5293, IMN, France
Laboratoire Franco-Israélien de Neurosciences, CNRS Bordeaux, France
daisuke.kase@u-bordeaux.fr

Thomas Boraud
Université de Bordeaux, CNRS UMR 5293, IMN, France
thomas.boraud@u-bordeaux.fr

Nicolas Rougier
INRIA Bordeaux Sud-Ouest, Bordeaux, France
Université de Bordeaux, CNRS UMR 5293, IMN, France
LaBRI, Université de Bordeaux, IPB, CNRS, UMR 5800, Talence, France
nicolas.rougier@inria.fr

Abstract

If basal ganglia are widely accepted to participate in the high-level cognitive function of decision-making, their role is less clear regarding the formation of habits. One of the biggest problem is to understand how goal-directed actions are transformed into habitual responses, or, said differently, how an animal can shift from an action-outcome (A-O) system to a stimulus-response (S-R) one while keeping a consistent behaviour.

We introduce a computational model (basal ganglia, thalamus and cortex) that can solve a simple two arm-bandit task using reinforcement learning and explicit valuation of the outcome (Guthrie et al. (2013)). Hebbian learning has been added at the cortical level such that the model learns each time a move is issued, rewarded or not. Then, by inhibiting the output nuclei of the model (GPi), we show how learning has been transferred from the basal ganglia to the cortex, simply as a consequence of the statistics of the choice. Because best (in the sense of most rewarded) actions are chosen more often, this directly impacts the amount of Hebbian learning and lead to the formation of habits within the cortex.

These results have been confirmed in monkeys (unpublished data at the time of writing) doing the same tasks where the BG has been inactivated using muscimol. This tends to show that the basal ganglia implicitely teach the cortex in order for it to learn the values of new options. In the end, the cortex is able to solve the task perfectly, even if it exhibits slower reaction times.

Keywords: basal ganglia, decision making, habits, Hebbian learning, reinforcement learning

1 Introduction

According to Schneider and Shiffrin (1977), a behaviour is automatic (i.e. becomes an habit) if a sensory events always elicit the same behaviour, even if the subject is doing something else. Think, for example, of someone entering a dark room while talking on the phone and switching on the light without ever really thinking about it. How this behaviour is acquired in the first place? How do we learn such habits? Seger and Spiering (2011), characterized habit learning using five definitional features: inflexible, incremental, unconscious, automatic, and insensitive to reinforcer devaluation. This tentative definition seems to be clearly in opposition with decision making that we could think of as flexible and highly sensitive to reinforcer devaluation. However, there are more and more evidences these two types of learning are somehow linked together (Yin and Knowlton (2006)), the question being how?

In a recent (unpublished) study, monkeys have been tested on a two-armed bandit task using a pharmacological approach, combining both decision making and procedural learning. Primates have been daily accustomed with the setup which is composed of four buttons placed on different directions (0, 90, 180 and 270) and another one on central position which detects the contact with monkeys hands. These buttons correspond to the four possible appearance directions of a cursor on a perpendicular screen. The monkeys initiated a trial keeping their hand on the central button, which induced the appearance of the cursor in the central position of the screen. After a random delay (0.5-1.5 s), two cues appeared in two (out of four) different positions determined randomly for each trial. Two experimental conditions were alternated by blocks of ten trials. On Habitual Condition (HC), the two cues (HC1 and HC2) are the ones with which the animals have been trained. Each cue had a fixed probability for monkeys to be rewarded (PHC1=0.75 and PHC2=0.25). The nature and the probability of each cue remained the same during each working session and between each working session. On Novelty Condition (NC), the two cues presented are new (NC1 and NC2). Each cue had a fixed probability for monkeys to be rewarded (PNC1=0.75 and PNC2=0.25). Once the cues are shown, the monkeys had a delay period (0.5-1.5s) to press the button according to one cue during a random time (0.5-1.5s). The cursor appeared on the chosen cue. An "end-of-trial" signal corresponding to the disappearance of the cursor indicated to the monkeys that the trial is actually finished. Monkeys were rewarded (0.3 ml of water) or not according to the reward probability of the chosen target. They could then start a new trial after an inter-trial interval included between 500ms and 1.5 ms. The procedure is summarised in 1 In one condition, the internal part of the Globus Pallidus (the main output structure of the BG) is injected with a saline solution (no effect) and in the other condition, it is injected with muscimol (inactivation). Results tend to show that performances related to familiar cues stay unchanged, independently of GPi inactivation, while learning of new cues is deeply impacted when GPi has been inactivated. This tends to suggest BG might be critical in learning decision and this learning can be later transferred to the cortex. In accordance with this hypothesis, we build a computational model whose architecture is described in the next section.

2 Methods

The architectural basis of the model has been originally described in Guthrie et al. (2013) where authors introduced a biophysical model of action selection that can solve a two-arm bandit task, such as the one describe above. Two parallel action selection pathways compose the model with inputs from distinct areas of the cortex: one for handling the cognitive action selection, and the other for the motor selection. The model includes the cortex (Cx), the thalamus (Th) and several nuclei of the BG: striatum (Str), the subthalamic nuclei (STN) and the globus pallidus internal and external (GPi, GPe). Each module is made of a closed- loop positive feedback direct pathway (Cx-Str-GPi-Th-Cx) and two closed-loop negative feedbacks, indirect pathway (Cx-Str-GPe-STN-GPi-Th-Cx) and hyperdirect pathway (Cx-STN-GPi-Th-Cx), and is based on the center-surround architecture of Mink (1996). The interactions between these three pathways are able to induce an action selection at the motor level. However, the task requires first the actual selection of the best cue before performing the corresponding motor action. In the Guthrie et al. (2013) model, this is implemented at the striatal level where a dopamine reward signal is used to implement a simple value-based learning. The general architectural of the model is illustrated on Figure 2.

In the original model, the inactivation of the basal ganglia output (GPi) results in the inability of the model to make a decision since there is no competitive mechanism at the cortical level. We thus added lateral connections in each cortical modules (self-excitation and surround inhibition) such that a unique cognitive and motor decision can be made. At this point, there is no guarantee that the motor decision corresponds to the cognitive ones. The model can choose a cue A but moves toward the location of a cue B. To overcome this problem, we also need to establish a cross-talking between the different cortical modules, independently of the BG pathways. This has been made using excitatory connections from and to the associative cortical module. Hebbian learning in cortical level allows the cortex to make a consistent decision in the absence of GPi, even if it does not guarantee to make the optimal decision.

One important property of the cortical decision is that it is significantly slower than the BG decision. This can be shown by measuring the time of motor decision; defined as the time required for the difference between the two most activated units in the motor cortex to become greater than a given threshold (40Hz). Before learning, the BG decision time (GPi is

intact) is around 250ms while the cortical decision time (GPi is disabled) is around 800 ms. This difference in timing is actually critical for the BG to teach the cortex as explained in the results section.

3 Results

We tested the model using 4 different paradigms:

- HC/GPi: Habitual condition using already learned stimuli with intact GPi.
- HC/NoGPi: Habitual condition using already learned stimuli with lesioned GPi.
- NC/GPi: Novel condition using non familiar stimuli with intact GPi.
- NC/NoGPi: Novel condition using non familiar stimuli with lesioned GPi.

Each condition has been tested for 250 experiments where each experiment consists in 120 consecutive trials (presentation of the cues, decision, potential reward and reset). Before starting an new experiment, the model is trained on the familiar stimuli until the performance is over 0.95. Performances were measured as the ratio of optimal choices compared to the number of trials. Response time has been recorded as the time of the motor decision. This time is relative to the stimulus onset (t=500ms).

As shown in 3a, our results are equivalent to the experiments in monkeys (3b). In the habitual condition, performances are optimal with or without lesion, indicating the cortex is able to make the optimal decision without the help of the basal ganglia. However, in the novel condition, things are quite different. For the intact model, the model starts a trial at chance level, giving random choices. Nevertheless, after a few trials (around 15), it reaches a near-optimal performance, indicating the model has learned the respective reward probability associated with each cue. For the lesioned model, the performances stay at the level of chance, indicating the cortex is unable to learn the task "on its own". It is to be noted that due to Hebbian learning, the lesioned model tend to first choose a given random cue and stick to this choice until the end of the experiment. If this is the right cue, the performance can reach 1 for a single experiment, but over the course of the 250 experiments, the mean performance is around 0.5.

4 Conclusion

The aim of this model is to gain a better understanding of the role of the basal ganglia in the formation of habits. It is based on a previous model by Guthrie et al. (2013) that explain the dynamic of action selection in the BG. The model has been further refined with connections at the cortical level which are consistent with neuro-anatomy. We also implemented Hebbian learning at the cortical level, independently of reward. However, since BG helped to choose the best action anytime, this results in having cortical learning to be naturally modulated according to the value of the different cue, simply because the best cue is chosen more often. After learning, the cortex is able to choose the best cue without help of the BG, hence forming a new habit.

References

M. Guthrie, A. Leblois, A. Garenne, and T. Boraud. Interaction between cognitive and motor cortico-basal ganglia loops during decision making: a computational study. *Journal of Neurophysiology*, 2013.

J. Mink. The basal ganglia: focused selection and inhibition of competing motor programs. *Progress in Neurobiology*, 1996.

W. Schneider and R.M. Shiffrin. Controlled and automatic human information processing: I. detection, search, and attention. *Psychological Review*, 1977.

C.A. Seger and B. J. Spiering. A critical review of habit learning and the basal ganglia. *Frontiers in Systems Neuroscience*, 2011.

H.H. Yin and B. J. Knowlton. The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, 2006.

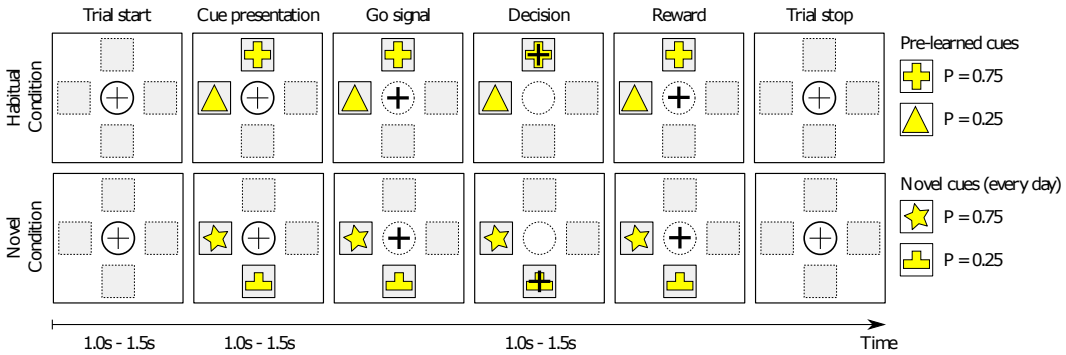


Figure 1: Behavioural paradigm. A session is made up of at least 250 trials broken down into alternate blocks of 10 trials in Habitual (top) or Novelty (bottom) Conditions. In each trial, two cues were displayed simultaneously in two out of four randomly chosen possible positions on the screen. The monkey signalled its choice by moving the cursor to one of the cues and was rewarded by 300 μl of fruit juice with a predefined fixed probability that depends on the choice. In the Habitual Conditions (top) the cues (HC1, P=0.75 and HC2, P=0.25) are the one with which the monkeys has been trained and therefore are familiar with. In the Novelty Condition (bottom), the cues (NC1 and NC2) have the same values (P=0,75 and P=0,25 respectively), but the pairs are changed (new shape and colours) for each session.

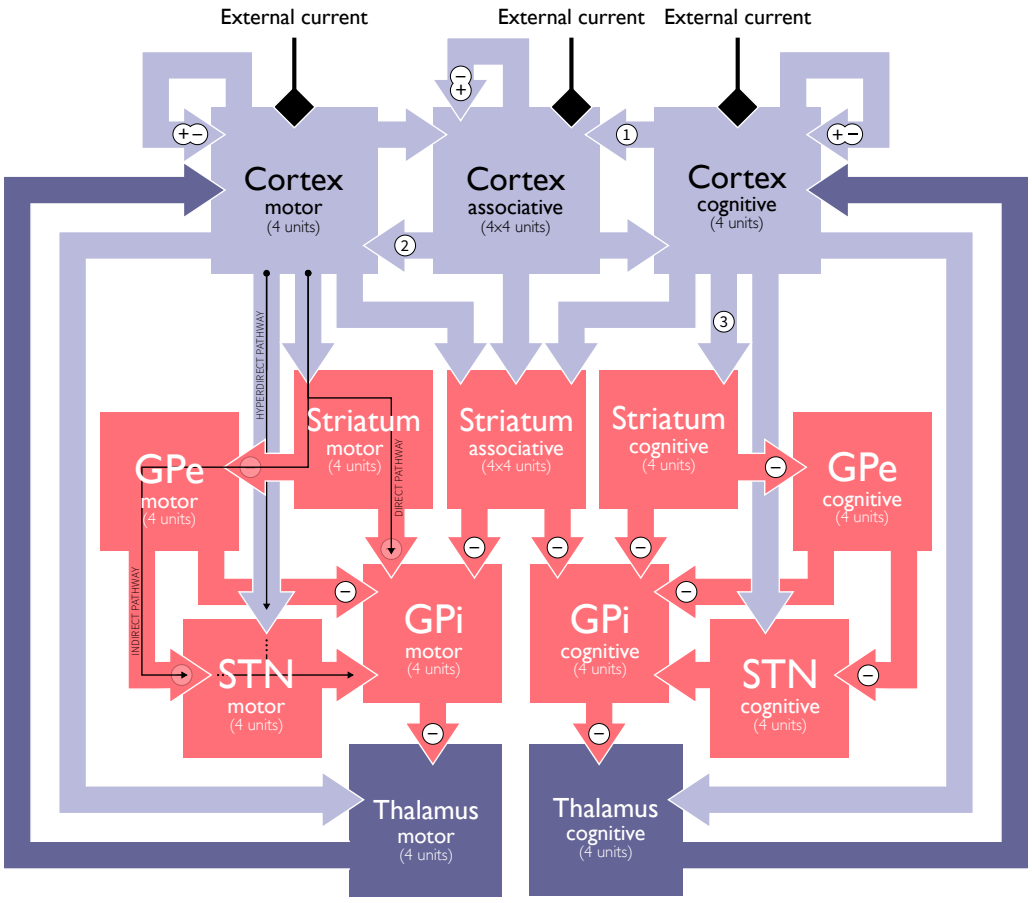
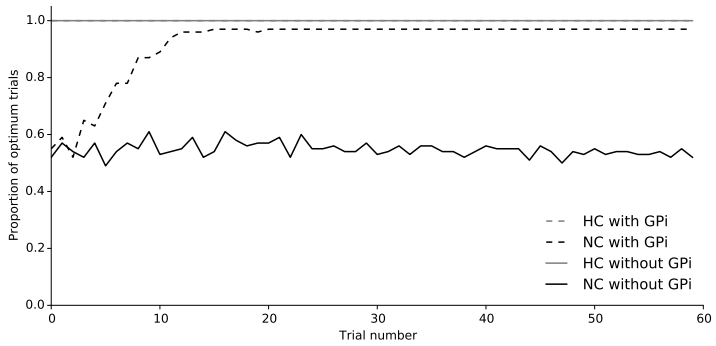
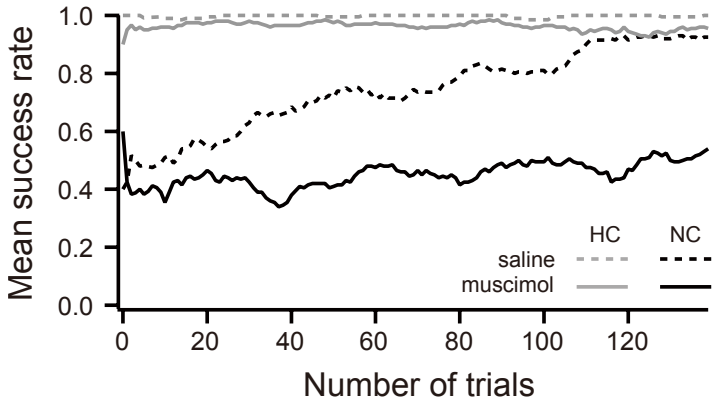


Figure 2: The main pathways in the model are the direct pathway (Cortex-Striatum-GPi-Thalamus-Cortex), the indirect pathway (Cortex-Striatum-GPe-STN-GPi-Thalamus-Cortex), and the hyperdirect pathway (Cortex-STN-GPi-Thalamus-Cortex). Learning occurs at two different levels: Hebbian learning from cognitive to associative cortex (1) and from associative to motor cortex (2), and reinforcement from cognitive cortex to cognitive striatum (3).



(a) Mean performances of model



(b) Mean performances of monkeys

Figure 3: Results averaged over 250 simulations. 3a) In HC, performances of the model are optimal (1), with or without GPi (the dashed line is not shown, because it coexists with the straight one). In NC, only the intact model (with GPi) is able to learn the new stimuli while lesioned model performances stay at the level of chance. 3b) Monkeys' performances are analogous to the models.

The Carli Architecture–Efficient Value Function Specialization for Relational Reinforcement Learning

Mitchell Keith Bloch
University of Michigan
Ann Arbor, MI, USA
bazald@umich.edu

John Edwin Laird
University of Michigan
Ann Arbor, MI, USA
laird@umich.edu

Abstract

We introduce Carli–a modular architecture supporting efficient value function specialization for relational reinforcement learning. Using a Rete data structure to support efficient relational representations, it implements an initially general hierarchical tile coding and specializes it over time using a fringe. This hierarchical tile coding constitutes a form of linear function approximation in which conjunctions of relational features correspond to weights with non-uniform generality. This relational value function lends itself to learning tasks which can be described by a set of relations over objects. These tasks can have variable numbers of both features and possible actions over the course of an episode and goals can vary from episode to episode. We demonstrate these characteristics in a version of Blocks World in which the goal configuration changes between episodes. Using relational features, Carli can solve this Blocks World task, while agents using only propositional features cannot generalize from their experience to solve different goal configurations.

1 Introduction

Many kinds of problems can be formulated as reinforcement learning problems. Simple reinforcement learning algorithms (such as both Sarsa(λ) and Q(λ)) directly suffer from the curse of dimensionality. The number of Q-values to be learned scales directly with the product of the size of the state-space, $|D_1 \times D_2 \times D_3 \times \dots|$, and the size of the action-space, $|\{A_1, \dots, A_n\}|$. As the state-space and action-space grow large for more complex problems, it becomes difficult to learn efficiently. The issue is a lack of some capacity to generalize.

Linear function approximation is one approach to adding the ability to generalize to these temporal difference methods. The simple value function can be replaced by n basis functions, $\phi_1, \dots, \phi_n$. The Q-function is estimated by $Q(s,a) = \sum_{i=1}^{n} \phi_i(s,a) w_i$, where $\phi_i(s,a)$ is commonly 1 or 0 to indicate whether the feature is currently active for a given state and action, or not, and where the weight, w_i, represents the value contribution of the feature for the action under consideration, or a partial Q-value. That each feature applies to many states provides the agent with the ability to learn about states which the agent has yet to visit. For example, the feature (*stack*, matches, *goal–stack*) applies to many actions throughout the state-space and not just to one specific state-action pair.

A second approach to support generalization from experience is to start with a coarse tile coding, and to break the tiles into smaller, more specific tiles over time. As the agent specializes its value function, it learns a more refined policy, but without the full cost associated with starting with a maximally specialized value function from the beginning. It is possible to combine this with linear function approximation by keeping coarser, more general tiles in the system and allowing general learning to continue even as the smaller, more specialized tiles are added [Bloch and Laird, 2013]. This strategy assumes that the hierarchy accurately captures the structure of the true value function to some degree, allowing more general patterns in the value function to be captured by the more general tiles and requiring the more specific tiles to capture the finer details.

Relational reinforcement learning is a third approach to support generalization from experience [Tadepalli *et al.*, 2004; Džeroski *et al.*, 2001]. Given relations and conjunctions thereof, it is possible to condense both the state-space and action-space for certain tasks. For example, if the agent is concerned that (*stack*, matches, *goal–stack*), it can learn about all actions in which that match is present, regardless of the labels of any of the blocks in either the *stack* or the *goal–stack*. As these features do not always depend on the size of the state-space and action-space, they can be particularly effective for scenarios with arbitrarily large state-spaces.

As relational reinforcement learning allows an agent to reason about objects more abstractly, it allows the agent to generalize from its experience to solve problems from outside its training set. The alternative of propositionalizing the relational representation tends to create far more features and has more difficulty transferring to similar problems.

We develop the Carli[1] architecture using all three of these methods to provide generalization. The intent is to explore not only how to efficiently implement a relational reinforcement learning agent with dynamically variable specialization, but to investigate how best to do this specialization and learn with this kind of system. The manner in which the Carli architecture maps states and actions to weights and specializes that mapping over time is critical. As explored by Bloch and Laird [2013], Whiteson *et al.* [2007], and McCallum [1996], Carli uses a tile coding that begins with large, general tiles and then specializes it over time to generate smaller tiles and more specific weights. As done by McCallum [1996], a fringe is stored alongside each leaf weight to allow the agent to make informed decisions about which features to use next when specializing a tile. However, instead of collecting instances, we store fixed-size metrics to be used by different specialization criteria. And like the system explored by Bloch and Laird [2013], it keeps more general tiles in the system as more specific ones are added, allowing more general learning to continue as the policy is refined. The major contribution of Carli is its unique use of the Rete algorithm to implement a computationally efficient value function, providing a specializable, hierarchical tile coding that supports relational features.

2 Related Work

Efficient rule matching is a well established problem in the context of rule-based systems. We map the problem of providing weights for linear function approximation onto this matching problem. The Soar cognitive architecture [Laird, 2012] and Soar-RL [Nason and Laird, 2004]–its implementation of reinforcement learning–were the primary inspiration for this work. Soar would require significant modification to support efficient value function specialization as implemented in Carli.

Driessens *et al.* [2001] implemented the RRL-TG algorithm. They implemented query packs as a way to use more memory to allow common parts of queries to be represented only once in their data structure. This optimization enforces a similar structure to that of Rete, sharing earlier tests in order to save work. RRL-TG requires specializations to be binary splits based on truth tests, which Carli does not require.

Adlin being the name of their Icarus-based architecture, Asgharbeygi *et al.* [2005] noted that "when the time constraint is extremely strict or the environment is changing too rapidly, Adlin does not provide a satisfying performance gain." The goal of their relational reinforcement learning system is to guide inference rather than to provide near optimal control, but one solution they considered to deal with their performance problem as part of their future work was "to incorporate the simple idea behind truth maintenance systems and Rete matchers," which is related to what we have done in our work.

In order to support online, relational reinforcement learning, it is necessary to support efficient detection of conjunctions of relational features specifically. A system explored by Bloch and Laird [2013] that used a trie to store the value function did not support the use of relational features. Their trie implementation was efficient but ill suited for supporting relational tests. It may have been possible to modify the implementation to support a variable number of actions, but any relational features would have needed to be be propositionalized. This propositionalization would have increased the number of features considerably and, due to the lack of a notion of variable binding, additionally made it difficult to support conjunctions of relational features that refer to the same variables.

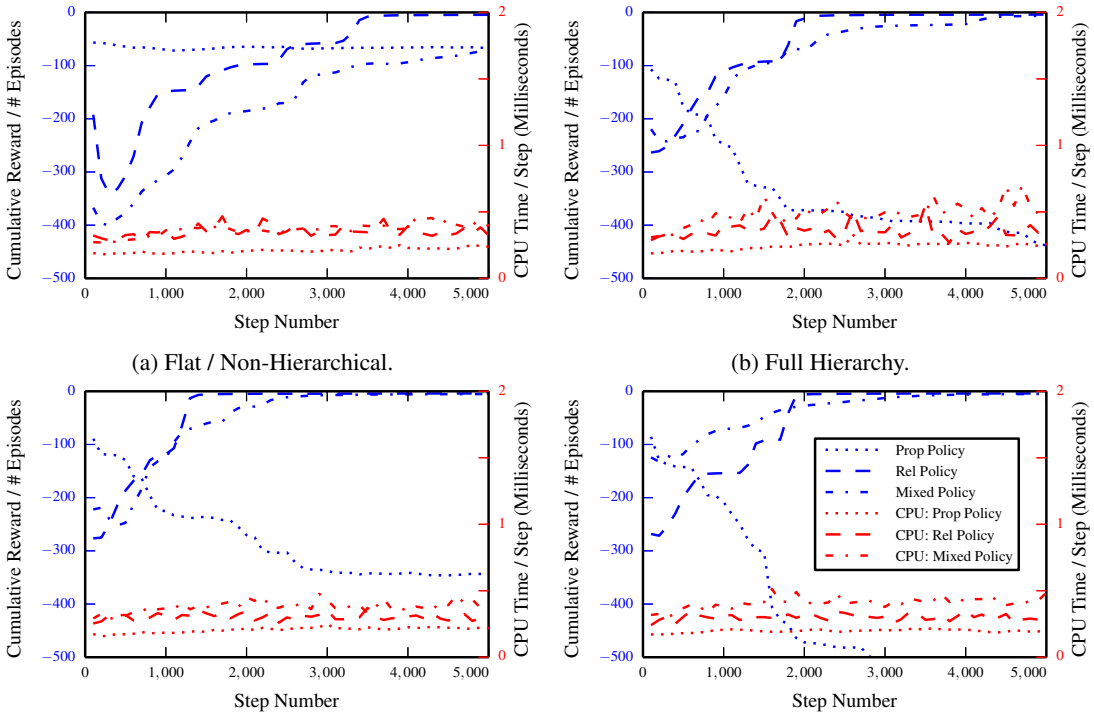
3 Carli

We use Rete [Forgy, 1979] to provide efficient rule matching and weight retrieval for our agents. Rete provides the relational structure, variable bindings, and efficiency we need to support relational features. Hierarchical tile codings are implemented in the Rete as a set of rules sharing the same tests up until feature specialization occurs, and the work to fire these shared tests is effectively free in the Rete. The cost of minimally firing and retracting rules to match changes in the environment is linear in the number of changes rather than in the size of the state-space. Efficient implementation of the Rete algorithm has been explored in detail by Doorenbos [1995].

One way to specialize a tile coding over time is to maintain a fringe of possible future specializations and to conditionally (or periodically) choose from among those possible specializations to expand the value function. As part of that expansion, it is necessary to create a new fringe for the new tiles which have been added to the tile coding.

The hierarchical structures of both Rete and this kind of tile coding line up in a way that makes their joint implementation very convenient, albeit non-trivial. We found it useful to define a rule grammar to specify the combinations of features. The rules adhering to this grammer together constitute our tile coding. Given our grammar, we can extract details about the types of features the rules are adding to the system. At the time of fringe expansion, we can separate the new features from the rules (in the case that they correspond to fringe tiles) and modify other rules to incorporate the new features in order to create a new fringe. Similarly, it is possible to collapse a tile coding to undo earlier specializations should it be deemed useful to do so.

[1] Carli can be downloaded from https://github.com/bazald/carli.



(a) Flat / Non-Hierarchical.

(b) Full Hierarchy.

(c) Value Criterion.

(d) Policy Criterion.

Figure 1: These plots display task performance (in blue) and total system CPU time (in red) for Carli agents in Blocks World. Propositional features are dotted and relational features are dashed. The legend in figure 1d applies to all figures 1a through 1d.

4 Blocks World

Blocks World presents a configuration of blocks, each with a unique label. The agent is tasked with moving one block at a time from the top of a stack of blocks to either the top of a different stack or the table. The agent completes the task when the blocks configuration matches the goal configuration.

The Blocks World task that interests us exposes the goal configuration to the agent and changes the goal from episode to episode. Agents using a standard propositional representation (dotted in figure 1) of this formulation of the Blocks World task should be unable to learn a near optimal strategy for solving new goal configurations because they must ultimately capture some details of the task specific to a particular goal configuration in order to function at all. Agents using relational representations (dashed in figure 1) should can capture connections between stacks of blocks in both the current configuration and the goal configuration, generalizing to different goal configurations. Agents using both representations together (dash-dotted in figure 1) should still be able to learn to solve the task, though they may require more time to do so.

The propositional features available to our agents are: whether the destination block is in the right place or not (a propositionalized relational feature that is equivalent to the second relational feature below); the name of the block to be moved; and the name of the destination block.

The relational features available to our agents are: whether the source stack of blocks matches a goal stack or not; a stack that is a subset of a goal stack would be considered matching; placing a block on top of such a stack would cause it to no longer match; whether the destination stack matches a goal stack or not; whether the top block of the source stack matches the top of the destination stack or not (a top block being considered to match the top of the destination stack if and only if the destination stack matches a goal stack and the additional block will allow it to continue to match); and whether the destination (stack) is the table.

We train our Carli agents on-policy with a discount rate of 0.9 and an eligibility trace decay rate of 0.3 using Blocks World instances with 4 blocks. Figures 1a through 1d plot the cumulative reward divided by the number of episodes solved and the current computational time required per step of the environment.

Figures 1a and 1b depict performance of agents doing all specialization before the first step. Both sets of agents specialize maximally before the first step, choosing features at random. The flat agents discard more general features from the system while the hierarchical agents keep the more general features for linear function approximation. It is clear that the hierarchical tile coding is advantageous to the agents that include the relational features, although it also allows divergence in the agent using only propositional features.

Figures 1c and 1d depict performance of agents doing online specialization after gaining some experience, but still early in their lifetimes. The agents using the value criterion attempt to minimize error when specializing the value function by choosing feature axes in the fringes which present large value discrepancies between Q-values. The agents using the policy criterion attempt to be more tolerant of error in the value function by choosing only feature axes that are likely to alter their policies when specializing the value function. These specialization criteria are derived from those presented by Whiteson *et al.* [2007]. The performance pattern for agents using the different sets of features predicted at the beginning of this section holds.

The memory utilization differs depending primarily on which features are included. The agents using only the propositional features tend to include approximately 52 weights in their value function. The agents using only the relational features tend to include approximately 25. And agents using both sets of features tend to include between 300 and 350 weights (304 for the value criterion, and 350 for the policy criterion, and 390 for the full hierarchy). As shown in the figures, the computational cost scales sublinearly with memory usage, although more selective criteria may be needed to keep memory costs down in large domains. Computing the relational features is more expensive than computing the propositional features, but their increased power is clearly valuable.

5 Future Work

We intend to explore additional domains in the future, including Infinite Mario [Togelius *et al.*, 2010]. However, the number of distinct variables is currently fixed by the features initially provided to the system. One extension would be to allow Carli agents to reason about a number of different blocks, enemies, and pits in the environment without enumerating a fixed number of block, enemy, and pit variables ahead of time. Its unclear whether this higher order fringe grammar would have sufficient value to be worth the complications involved in its implementation.

Adding support for Greedy-GQ(λ) [Maei, 2011], a temporal difference method that is guaranteed to converge when using linear function approximation, would probably improve the performance of Carli agents.

We could benefit from better tests to determine which features to include, in addition to our value and policy criteria. There is a tradeoff between specializing quickly and specializing slowly. If our system specializes more quickly it can begin learning a more specialized policy faster, but it may be more likely to geneate a suboptimal Rete structure.

Our Rete value function implementation supports efficient despecialization to undo suboptimal specializations, but we have yet to develop criteria to allow us to decide when to despecialize or respecialize with different features tests higher up in the Rete.

References

[Asgharbeygi *et al.*, 2005] Nima Asgharbeygi, Negin Nejati, Pat Langley, and Sachiyo Arai. Guiding inference through relational reinforcement learning. In *Proceedings of the 15th International Conference on Inductive Logic Programming*, ILP'05, pages 20–37, Berlin, Heidelberg, 2005. Springer-Verlag.

[Bloch and Laird, 2013] Mitchell Keith Bloch and John Edwin Laird. Online value function improvement. In *RLDM 2013: Extended Abstracts*, pages 111–115, 2013.

[Doorenbos, 1995] Robert B. Doorenbos. *Production Matching for Large Learning Systems*. PhD thesis, Pittsburgh, PA, USA, 1995. UMI Order No. GAX95-22942.

[Driessens *et al.*, 2001] Kurt Driessens, Jan Ramon, and Hendrik Blockeel. Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. In *Machine Learning: ECML 2001*, pages 97–108. Springer, 2001.

[Džeroski *et al.*, 2001] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.

[Forgy, 1979] Charles Lanny Forgy. *On the Efficient Implementation of Production Systems.* PhD thesis, Pittsburgh, PA, USA, 1979. AAI7919143.

[Laird, 2012] John E. Laird. *The Soar Cognitive Architecture*. The MIT Press, 2012.

[Maei, 2011] Hamid Reza Maei. *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, 2011.

[McCallum, 1996] Andrew Kachites McCallum. *Reinforcement learning with selective perception and hidden state*. PhD thesis, 1996. AAI9618237.

[Nason and Laird, 2004] Shelley Nason and John E. Laird. Soar-rl: Integrating reinforcement learning with soar. In *Cognitive Systems Research*, pages 51–59, 2004.

[Tadepalli *et al.*, 2004] Prasad Tadepalli, Robert Givan, and Kurt Driessens. Relational reinforcement learning: An overview. In *Proceedings of the ICML-2004 Workshop on Relational Reinforcement Learning*, pages 1–9, 2004.

[Togelius *et al.*, 2010] Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. The 2009 mario ai competition. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[Whiteson *et al.*, 2007] Shimon Whiteson, Matthew E. Taylor, and Peter Stone. Adaptive tile coding for value function approximation, 2007.

A Deeper Look at Planning as Learning from Replay

Harm van Seijen
Department of Computing Science
University of Alberta
Edmonton, Alberta
T6G 2E8, Canada
harm.vanseijen@ualberta.ca

Richard S. Sutton
Department of Computing Science
University of Alberta
Edmonton, Alberta
T6G 2E8, Canada
sutton@cs.ualberta.ca

Abstract

In reinforcement learning, the notions of experience replay, and of planning as learning from replayed experience, have long been used to find good policies with minimal training data. Replay can be seen either as model-based reinforcement learning, where the store of past experiences serves as the model, or as a way to avoid a conventional model of the environment altogether. In this paper, we look more deeply at how replay blurs the line between model-based and model-free methods. Specifically, we show for the first time an exact equivalence between the sequence of value functions found by a model-based policy-evaluation method and by a model-free method with replay. We then use insights gained from these relationships to design a new reinforcement learning algorithm for linear function approximation. This method, which we call *forgetful LSTD(λ)*, improves upon regular LSTD(λ) because it extends more naturally to online control, and improves upon linear Dyna because it is a multi-step method, enabling it to perform well even in non-Markov problems or, equivalently, in problems with significant function approximation.

Keywords: reinforcement learning, model-based learning, experience replay, LSTD(λ)

Acknowledgements

The authors thank Joseph Modayill, Kavosh Asadi, Hado van Hasselt and Rupam Mahmood for insights and discussions contributing to the results presented in this paper. We also gratefully acknowledge funding from Alberta Innovates — Technology Futures and from the Natural Sciences and Engineering Research Council of Canada.

1 Introduction

In reinforcement learning (RL) (Sutton & Barto, 1998), value-function based methods are traditionally divided into two categories: model-free and model-based. Model-free, or direct, methods learn a value function directly from samples, from which a policy can be easily derived. In contrast, model-based, or indirect, methods learn a model of the environment dynamics and derive a value function or policy from the model using planning. Compared to model-free methods, model-based methods are typically more computational expensive, but make more efficient use of observed experience. Interestingly, neural and behavioral data suggests that the brain also has multiple reasoning systems that trade-off computational efficiency for sample efficiency in different ways (Daw et al., 2005).

While many RL methods can be easily classified as either model-free or model-based, there are also methods for which this is less straightforward. For example, LSTD (Bradtke & Barto, 1996; Boyan, 2002), which computes a least-squares solution over a set of samples, can be considered model-free because it does not explicitly construct a model of the environment dynamics. However, its characteristics, high computational cost combined with an efficient use of samples, are typical of those of a model-based method. This also holds — albeit to a lesser extend — for model-free methods enhanced by experience replay (Lin, 1992). Hence, some people argue that a sample set should also be considered a model. Another method that is hard to classify is linear Dyna (Sutton et al., 2008), which uses observed samples to update a value function, *as well as* construct a model, which is then used to produce simulated samples to further improve the value function.

The distinction between model-free and model-based methods is further blurred by the theoretical result that the fixed point of a linear, least-squares model on a set of samples is the same as the LSTD solution of those samples (Sutton et al., 2008; Parr et al., 2008). In practise, however, the value function computed by a model-based method (at any particular time) is different. This is partly because it learns only an estimate of the least-squares model, and because, due to bounded computation per time step, its iterative planning process terminates before convergence is reached.

In this paper, we show that the relation between model-free and model-based methods runs even deeper than assumed up to now. Specifically, we show for the first time an exact equivalence between the sequence of value functions found by a model-based method and by a model-free method with replay. This result raises an intriguing question: is there really a fundamental difference between model-free and model-based methods, or are they just two different ways of looking at the same thing?

The replay technique we use is different from the classical technique introduced by Lin (1992). Rather than presenting old samples as new to the learning agent, we replay updates with an improved update target, starting from the initial value estimate. This way of doing replay yields a powerful strategy for deriving methods that trade off computational efficiency with sample efficiency in different ways. We demonstrate this by using it to derive a new method, which we call *forgetful LSTD(λ)*. This method gracefully forgets old information when new information is observed (similarly to linear Dyna), which is important in control tasks. On the other hand, it uses a multi-step model similar to the model used by LSTD(λ) (Boyan, 2002). Using multi-step models is key for obtaining robust model-based behaviour in problems with significant function approximation, or similarly, state aggregation. However, this is not widely known, as evidenced by a number of recent publications that conclude that learned models appear to be fundamentally limited, based on observed behaviour from a one-step model. We illustrate the importance of multi-step models with a small experiment.

2 Problem Setting and Notation

We focus in this paper primarily on discrete-time *Markov reward processes* (MRPs), which can be described as 4-tuples of the form $\langle \mathcal{S}, p, r, \gamma \rangle$, consisting of $\mathcal{S}$, the set of all states; $p(s'|s)$, the transition probability function, giving for each state $s \in \mathcal{S}$ the probability of a transition to state $s' \in \mathcal{S}$ at the next step; $r(s, s')$, the reward function, giving the expected reward after a transition from s to s'. γ is the discount factor, specifying how future rewards are weighted with respect to the immediate reward.

The value-function v of an MRP maps each state $s \in \mathcal{S}$ to the expected value of its *return*, which is the discounted sum of rewards following that state:

$$v(s) = \mathbb{E}\Big\{ \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i} \,|\, S_t = s \Big\}.$$

We consider the case where the learner does not have access to s directly, but can only observe a feature vector $\boldsymbol{\phi}(s) \in \mathcal{R}^n$. The value function is approximated using linear function approximation, in which case the value of a state is the inner product between the weight vector $\boldsymbol{\theta} \in \mathcal{R}^n$ and its feature vector: $\hat{v}_t(s) = \boldsymbol{\theta}_t^\top \boldsymbol{\phi}(s)$. As a shorthand, we will indicate $\boldsymbol{\phi}(S_t)$, the feature vector of the state visited at time step t, by $\boldsymbol{\phi}_t$.

The basic definition of an MRP is often extended with the concept of *terminal states*, which terminate the return and divide the environment interaction into episodes. We model this by using a time-dependent discount factor that is equal to 0 when a terminal state is reached. This way of modelling episodic tasks makes that we can treat episodic tasks and continuing tasks fully uniformly.

3 Planning by Replaying the Past

Classical experience replay, as introduced by Lin (1992), maintains a bag of (recent) samples that are used for additional updates. Basically, the sequence of observed samples is artificially enlarged by injecting old samples. Replaying experience in this way is a simple strategy to improve sample efficiency, but it also has some downsides. For example, it cannot be combined with multi-step update targets such as the λ-return (Sutton, 1988), because old samples are mixed with new samples. We introduce an alternative form of replaying samples that can be combined with any update target.

Our way of replaying experience is illustrated by Algorithm 1. Starting from the initial weight vector, all previously performed updates are redone —in the same order— with an update target based on the most recent weight vector. This update sweep can be repeated multiple times per time step (controlled by the K-parameter). This form of replaying experience also forms the basis of best-match learning, which combines a model-free value function with a partial, tabular model to obtain convergence in the tabular case (van Seijen et al., 2011). It is also related to the idea behind fitted Q-iteration (Riedmiller, 2005).

Algorithm 1: Replaying TD(0) updates

INPUT: $\alpha, K, \boldsymbol{\theta}_{init}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}$
$\mathcal{V} \leftarrow \emptyset$ ($\mathcal{V}$ *is an ordered set)*
obtain initial $\boldsymbol{\phi}$
Loop:
  obtain next feature vector $\boldsymbol{\phi}'$, γ and reward R
  add $(\boldsymbol{\phi}, R, \gamma, \boldsymbol{\phi}')$ to $\mathcal{V}$
  Repeat K times:
    $\boldsymbol{\theta}_{target} \leftarrow \boldsymbol{\theta}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}$
    For all $(\boldsymbol{\varphi}, r, \bar{\gamma}, \boldsymbol{\varphi}')$ in $\mathcal{V}$ (from oldest to newest):
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[r + \bar{\gamma}\,\boldsymbol{\theta}_{target}^\top \boldsymbol{\varphi}' - \boldsymbol{\theta}^\top \boldsymbol{\varphi}\big]\boldsymbol{\varphi}$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$

Algorithm 2: Planning with a learned linear model

INPUT: $\alpha, K, \boldsymbol{\theta}_{init}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}$
$\boldsymbol{b} \leftarrow \boldsymbol{\theta}_{init}$, $F \leftarrow \mathbf{0}$
obtain initial $\boldsymbol{\phi}$
Loop:
  obtain next feature vector $\boldsymbol{\phi}'$, γ and reward R
  $F \leftarrow F + \alpha\big[\gamma\boldsymbol{\phi}' - F\boldsymbol{\phi}\big]\boldsymbol{\phi}^\top$
  $\boldsymbol{b} \leftarrow \boldsymbol{b} + \alpha(R - \boldsymbol{b}^\top\boldsymbol{\phi})\boldsymbol{\phi}$
  Repeat K times:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{b} + F^\top\boldsymbol{\theta}$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$

Algorithm 2 shows a model-based method. It uses the same model, and model update rules, as linear Dyna. However, instead of using the model to generate simulated samples, the model is used to perform planning updates (similar to value iteration updates for a tabular model). The theorem below states that the sample-based approach of Algorithm 1 and the model-based approach of Algorithm 2 are, in fact, equivalent.

Theorem 1 *Given the same sequence of samples and parameter settings, Algorithm 1 and Algorithm 2 compute the same weight vectors at each time step.*

Proof The outline of the proof is to rewrite the updates from Algorithm 1 in terms of matrices and vectors, which results in the F and $\boldsymbol{b}$ updates of Algorithm 2.

Let $\{\boldsymbol{\phi}_0, R_1, \gamma_1, \boldsymbol{\phi}_1, \ldots, R_t, \gamma_t\boldsymbol{\phi}_t\}$ be the observed experience sequence at time t. In addition, let $\boldsymbol{\theta}_{(t)}$ be the result of a single replay sweep over the t samples in $\mathcal{V}$. In other words, $\boldsymbol{\theta}_{(t)}$ is incrementally defined by:

$$\boldsymbol{\theta}_{(i+1)} = \boldsymbol{\theta}_{(i)} + \alpha\,[R_{i+1} + \gamma(\boldsymbol{\theta}_{target})^\top\boldsymbol{\phi}_{i+1} - \boldsymbol{\theta}_{(i)}^\top\boldsymbol{\phi}_i]\boldsymbol{\phi}_i\,, \qquad \text{for } 0 \leq i < t$$

with $\boldsymbol{\theta}_{(0)} := \boldsymbol{\theta}_{init}$. This update can be rewritten as:

$$\boldsymbol{\theta}_{(i+1)} = (\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top)\,\boldsymbol{\theta}_{(i)} + \alpha\boldsymbol{\phi}_i R_{i+1} + \alpha\gamma_{i+1}\boldsymbol{\phi}_i\boldsymbol{\phi}_{i+1}^\top\boldsymbol{\theta}_{target} \tag{1}$$

where $\mathcal{I}$ is the identity matrix. Now, consider that the following holds:

$$\boldsymbol{\theta}_{(i)} = \boldsymbol{b}_i + F_i^\top\boldsymbol{\theta}_{target}\,, \tag{2}$$

then

$$\begin{aligned}\boldsymbol{\theta}_{(i+1)} &= (\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top)\,(\boldsymbol{b}_i + F_i^\top\boldsymbol{\theta}_{target}) + \alpha\boldsymbol{\phi}_i R_{i+1} + \alpha\gamma_{i+1}\boldsymbol{\phi}_i\boldsymbol{\phi}_{i+1}^\top\boldsymbol{\theta}_{target}\\ &= b_{i+1} + F_{i+1}^\top\boldsymbol{\theta}_{target}\end{aligned}$$

with

$$\boldsymbol{b}_{i+1} = (\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top)\boldsymbol{b}_i + \alpha\boldsymbol{\phi}_i R_{i+1} \tag{3}$$

$$F_{i+1}^\top = (\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top)F_i^\top + \alpha\gamma_{i+1}\boldsymbol{\phi}_i\boldsymbol{\phi}_{i+1}^\top\,. \tag{4}$$

So, if (2) holds for i, and $F^\top$ and $\boldsymbol{b}$ are updated according to (3) and (4), then (2) holds for $i+1$. It can be easily checked that (2) is true for $i = 0$, if $\boldsymbol{b}_0 = \boldsymbol{\theta}_{init}$ and $F_0 = \mathbf{0}$. Hence, with this initialization, it is true for all i. Finally, updates (3) and (4) can be rewritten as:

$$\begin{aligned} \boldsymbol{b}_{i+1} &= \boldsymbol{b}_i + \alpha\big[R_{i+1} - \boldsymbol{b}_i^\top \boldsymbol{\phi}_i\big]\boldsymbol{\phi}_i \\ F_{i+1} &= F_i + \alpha\big[\gamma_{i+1}\boldsymbol{\phi}_{i+1} - F_i\boldsymbol{\phi}_i\big]\boldsymbol{\phi}_i^\top \,, \end{aligned}$$

which are the model updates of Algorithm 2. ∎

4 Forgetful LSTD(λ)

In this section, we introduce a new method that combines a multi-step model, similar to the one of LSTD(λ), with graceful forgetting, as occurs in linear Dyna.

4.1 Derivation (outline)

Applying the replay strategy displayed in Algorithm 1 to the true online TD(λ) algorithm (van Seijen & Sutton, 2014), and rewriting the updates in terms of vectors and matrices, just like we did in the proof of Theorem 1, results in the following updates:

$$\boldsymbol{b}_{i+1} = \big(\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top\big)\boldsymbol{b}_i + \alpha\boldsymbol{e}_{i+1}R_{i+1} \tag{5}$$

$$F_{i+1}^\top = \big(\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top\big)F_i^\top + \alpha\boldsymbol{e}_{i+1}[\gamma_{i+1}\boldsymbol{\phi}_{i+1} - \boldsymbol{\phi}_i]^\top + \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top \,, \tag{6}$$

with $\boldsymbol{b}_0 = \boldsymbol{\theta}_0$ and $F_0^\top = \mathbf{0}$, and

$$\boldsymbol{e}_{i+1} = \big(\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top\big)\gamma_i\lambda\boldsymbol{e}_i + \boldsymbol{\phi}_i \,,$$

with $\boldsymbol{e}_0 = \mathbf{0}$. Note that for $\lambda = 0$, $\boldsymbol{e}_{i+1} = \boldsymbol{\phi}_i$ and equations (5) and (6) reduce to equations (3) and (4).

Using equations (5) and (6), we could generalize Algorithm 2 such that it uses a multi-step model instead of a one-step model. Instead, we rewrite the equations, using $A_i := (\mathcal{I} - F_i^\top)/\alpha$ and $\boldsymbol{d}_i := \boldsymbol{b}_i/\alpha$ (for all i), as:

$$\begin{aligned} \boldsymbol{d}_{i+1} &= \big(\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top\big)\boldsymbol{d}_i + \boldsymbol{e}_{i+1}R_{i+1} \\ A_{i+1} &= \big(\mathcal{I} - \alpha\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top\big)A_i + \boldsymbol{e}_{i+1}\left[\boldsymbol{\phi}_i - \gamma_{i+1}\boldsymbol{\phi}_{i+1}\right]^\top \,, \end{aligned}$$

with $\boldsymbol{d}_0 = \boldsymbol{\theta}_{init}/\alpha$ and $A_0 = \mathcal{I}/\alpha$. Finally, rewriting Equation (2) in terms of A_i and $\boldsymbol{d}_i$ yields:

$$\boldsymbol{\theta}_{(i)} = \boldsymbol{\theta}_{target} + \alpha(\boldsymbol{d}_i - A_i\boldsymbol{\theta}_{target}) \,.$$

The pseudocode implementing these updates is shown in Algorithm 3. We added two generalizations: we allow the matrix A and vector $\boldsymbol{d}$ to be initialized randomly, and we allow the step-size used in the update of the model to be different than the step-size used in the planning updates. The former, we indicate by β; the latter by α. The reason for making this distinction is that α and β influence the computed value functions in very different ways. The parameter β is a forgetting parameter that determines how easily old information is overwritten by new information. It directly influences the model and hence the fixed point. On the other hand, α is a parameter that influences the iterative process for finding this fixed point. Its value determines if, and how quickly, this process converges.

We call this method forgetful LSTD(λ) for obvious reasons: when the forgetting parameter β is set equal to 0, and $A_{init} = \mathbf{0}$ and $\boldsymbol{d}_{init} = \mathbf{0}$, the model A_t, $\boldsymbol{d}_t$ reduces to the model learned by LSTD(λ) (Boyan, 2002). Note that LSTD(λ) traditionally solves the model using the inverse of A_t, whereas Algorithm 3 uses an iterative process. While in principle these techniques can be interchanged, an iterative process offers more flexibility and control over computation time, which is important in an online setting.

Algorithm 3: Forgetful LSTD(λ)

INPUT: $\alpha, \beta, \lambda, K, \boldsymbol{\theta}_{init}, \boldsymbol{d}_{init}, A_{init}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}$, $\boldsymbol{d} \leftarrow \boldsymbol{d}_{init}$, $A \leftarrow A_{init}$
obtain initial $\boldsymbol{\phi}$
$\boldsymbol{e} \leftarrow \mathbf{0}$
Loop:
  obtain next feature vector $\boldsymbol{\phi}'$,γ and reward R
  $\boldsymbol{e} \leftarrow (\mathcal{I} - \beta\boldsymbol{\phi}\boldsymbol{\phi}^\top)\boldsymbol{e} + \boldsymbol{\phi}$
  $A \leftarrow (\mathcal{I} - \beta\boldsymbol{\phi}\boldsymbol{\phi}^\top)A + \boldsymbol{e}(\boldsymbol{\phi} - \gamma\boldsymbol{\phi}')^\top$
  $\boldsymbol{d} \leftarrow (\mathcal{I} - \beta\boldsymbol{\phi}\boldsymbol{\phi}^\top)\boldsymbol{d} + \boldsymbol{e}R$
  $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e}$
  Repeat K times:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\boldsymbol{d} - A\boldsymbol{\theta})$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$

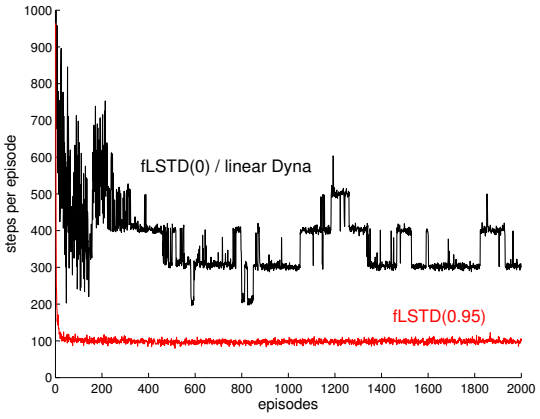


Figure 1: Multi-step versus one-step model.

4.2 Extension to Control

One of the main advantages of having a prediction method that gracefully forgets old information when new information comes in is that it can be used for control with minimal modification. The feature vectors observed simply have to be features describing state-action pairs, $\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t, A_t)$, instead of only states. With these features, values can be computed that predict the return for separate actions in a state (i.e., the *Q-values*), which can then be used for any exploration strategy, for example ϵ-greedy or some *softmax* selection strategy.

The role of forgetting is vital in a control setting, because the policy changes over time. Hence, old samples are no longer representative for the current policy and can inhibit policy improvement when maintained in the model.

4.3 Multi-Step Model versus One-Step Model

To demonstrate the importance of a multi-step model in problems with significant function approximation we perform an empirical comparison between forgetful LSTD(λ) with $\lambda = 0$ and $\lambda = 0.95$, on the mountain car control task (Sutton & Barto, 1998) We set $\beta = \alpha$, $\boldsymbol{d}_{init} = \boldsymbol{\theta}_{init}/\alpha$ and $A_{init} = \mathcal{I}/\alpha$. With these settings, forgetful LSTD(λ) with $\lambda = 0$ computes the same value functions as Algorithm 2 (as well as Algorithm 1), which is based on the (one-step) linear Dyna model.

Our mountain car task implementation is as described by Sutton & Barto (1998), but we make the task more challenging by using coarse tile coding: 3 tilings, each consisting of 3-by-3 tiles. We normalize the feature vectors, and use $\alpha = 0.01$ and $K = 1$. We used ϵ-greedy exploration with $\epsilon = 0.01$. The maximum episode length is set at 10,000 steps.

Figure 1 shows the performance over the first 2000 episodes, averaged over 100 independent runs. Note that the graph of linear Dyna contains performance jumps with a size of roughly 100 steps, which corresponds with 1 run out of the 100 runs hitting the 10,000 steps limit. This illustrates very clearly the stability issues that one-step models can cause.

5 Conclusion & Future Work

We showed that the relation between sample-based and model-based approaches runs much deeper than previously assumed. Specifically, we showed that TD(0) enhanced by replay computes, at each time step, the same values as a method based on a linear model. This result proves that these seemingly different approaches are simply different ways of looking at the same updates. To obtain this result, we used a form of replay that, is contrast to the classical way, can be combined with any update target. We demonstrated this by deriving a new, sample-efficient method, that can be used for control in tasks with significant function approximation. This method is just one example of how this new insight can be exploited. In future work, we intend to derive new methods that apply this idea to non-linear function approximation, as well as methods that can flexibly adapt to the available amount of computational resources by using partial models.

References

Boyan, J. A. Least-squares temporal difference learning. *Machine Learning*, 49:233–246, 2002.

Bradtke, S. J. and Barto, A. G. Linear least-squares algorithms for temporal-difference learning. *Machine Learning*, 22: 33–57, 1996.

Daw, N. D., Niv, Y., and Dayan, P. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8:1704–1711, 2005.

Lin, L. J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8: 293–321, 1992.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M.L. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pp. 752–759, 2008.

Riedmiller, M. Neural fitted Q iteration first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European conference on machine Learning*, 2005.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.

Sutton, R. S., Szepesvári, C., Geramifard, A., and Bowling, M. Dyna-style planning with linear function approximation and prioritized sweeping. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 528–536, 2008.

van Seijen, H., Whiteson, S., van Hasselt, H., and Wiering, M. Exploiting best-match equations for efficient reinforcement learning. *Journal of Machine Learning Research*, 12:2045–2094, 2011.

van Seijen, H. H. and Sutton, R. S. True online TD(λ). In *Proceedings of the 31th international conference on Machine learning (ICML)*, 2014.

Lost causes and unobtainable goals: Dynamic choice behavior in multiple goal pursuit

Jason L. Harman
Social and Decision Sciences
Carnegie Mellon University
jharman@cmu.edu

Claudia Gonzalez-Vallejo
Department of Psychology
Ohio University
gonzalez@ohio.edu

Jeffery B. Vancouver
Department of Psychology
Ohio University
vancouv@ohio.edu

Abstract

Lost causes and unobtainable goals: Dynamic choice behavior in multiple goal pursuit.

How do people choose to split their time between multiple pursuits when one of those pursuits becomes unobtainable? Can people cut their losses or do they more often chase a lost cause? We created a procedure where participants make repeated decisions, choosing to spend their free time between three different domains. One of these domains was a lost cause or unobtainable goal that would require all of the participant's resources to maintain. We found that participants will chase a lost cause, or continue to commit resources to a domain, despite harming other domains, but only when that domain is considered the most important of the three. When the lost cause is not the most important of the three domains participants will cut their losses deescalating their commitment to that domain.

Keywords: Goal Pursuit, Resource Allocation

Acknowledgements

1 Introduction

In the current studies, we examine how people allocate their time among different pursuits over time when one pursuit is an unobtainable goal. We varied the domains/activities available as well as their value and importance. In general, rational decision making considerations would entail cutting losses to free up time to devote to other pursuits. In fact, the ability to disengage from an unobtainable goal has been shown to be positively related wellbeing (Wrosch et al., 2003). However, the research on sunk costs (Arkes and Blumer, 1985) and keeping options (Shin and Arieley, 2004) open suggests that people may devote more resources to the unobtainable domain, essentially chasing a lost cause. There are three important aspects of the current work that have not been studied together: explicit opportunity costs, repeated choices with feedback regarding the results of the decisions, and using domains other than monetary choices allowing for differences in a priori importance to be measured.

2 Experiment 1

To explore time allocation decisions, we created a computer game we called Sim Life (see Figure 1) where participants made repeated choices to spend their free time among three domains. The status on each domain (i.e., the cumulative rewards or losses) is displayed throughout the task; the status of a domain improves when it is selected and decays when not chosen. Participants made 100 choices, simulating the number of days in an academic quarter. After each choice participants were shown a five second slide show of pictures representing the domain they had just selected to simulate the passage of time. The status of each domain was calculated on a 100 point scale with the visual feedback composed of ten categorical levels. The feedback functions we employed were structured so that two domains decayed by 3 points every trial and were increased by 7 points if chosen. The third option (the lost cause) either decayed faster (6 points per trial) or increased by a smaller increment if chosen (4 points).

Before starting the game, participants indicated their personal importance of each domain in three separate questions. The first was a rank ordering of the three domains. Second they indicated on a 1-5 scale how important each domain was to them. Finally, participants allocated 100 points between the three domains representing their relative importance. Results from these three measures were consistentthroughout.

In the game, participants were told that they have a set schedule of classes, studying, and work which leaves them two hours of free time each day that they can choose to spend on one of the three domains. In Experiment 1, the three domains were friends, their romantic partner, and academics. In addition to the reason for the lost cause (weak reward, strong loss), our other between-subjects condition was which domain was instantiated as the lost cause. Based on pilot data, we expected academics to be considered more important than the other two domains.

2.1 Methods

59 (37 F, mean age = 20.55) participants were randomly assigned to one of three conditions. The only difference between conditions was which domain was instantiated as the lost cause; relationship, friends, or academics.

2.2 Results

Mean importance ratings. For simplicity, we only present results of paired samples *t* tests using the 1-5 importance ratings using a Bonferoni correction. Analyses of the other measures are consistent with the reported results. Consistent with pilot studies, participants rated academics as more important than relationship (t (58) = 7.28, $p < .01$) and friends (t (58) = 8.23, $p < .01$). There was no difference between relationship and friends (t (58) = 0.146, $p = .88$).

Choice. To analyze choice over time, we split the data into 10 sequential blocks of trials, calculating the choice frequency for each domain in each block. Mean choice proportions for each condition are shown in Figure 1 with the domain instantiated as a lost cause highlighted. To examine choice behavior, individual change scores across the ten blocks of trials were calculated for each participant in each domain with negative scores indicating de-escalation of resource allocation across trials, and positive scores indicating increased resource allocation. To test whether choices between the lost cause and the other two domains differed, we performed a repeated measures ANOVA on the change scores from each domain across the three conditions revealing a significant interaction between domain change scores and which domain was the lost cause (F(4, 120) = 2.658, $p < .05$, *partial eta*^{2} = .081).

When academics (the most important domain) was the lost cause, participants escalated their resource allocation to academics as trials progressed, chasing a lost cause (mean change score = 2.03, SD = 2.83, significantly different than 0, t (30) = 3.99, $p < .01$). When the friends domain was the lost cause, participants cut their losses for friends, allocating fewer choices to the domain as trials progressed on average (mean change score = -1, SD = 2.68) though this was not significantly different than 0 (t (15) = 1.49, $p = 1.57$). When relationships was the lost cause, participants

neither escalated nor de-escalated resource allocation to that domain across trials (mean change score = 0.1765, SD = 2.5, t (15) = 1.49, p = 1.57).

The status of each domain is a direct function of choice frequency (given the feedback structure). When participants were able to cut their losses, that is stop selection of domains that were not advantageous (friends and relationships), they did better overall and had M = 2.05 in each of the lost cause domains and an overall mean of 32.45 across all domains. In contrast, when the lost cause was in the domain they cared most about (academics) they chased the lost cause earning fewer points altogether with academics getting M = 24.4 and M = 11.96 across all three domains which is significantly lower than the mean final status of the other two groups (t (58) = 3.17, p < .01, d = .8196).

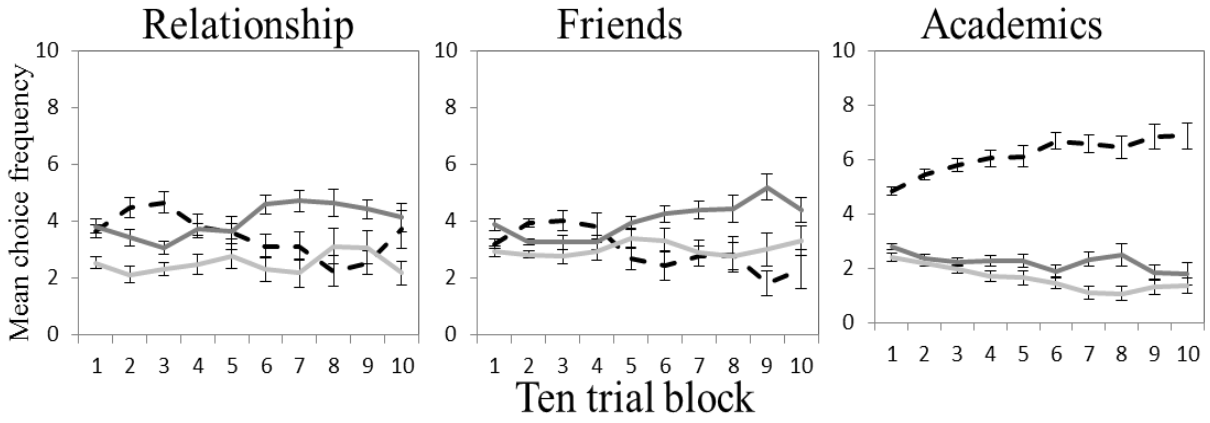


Figure 1. Experiment 1 choice frequency. The mean choice frequency across block of ten trials for each domain is displayed for each condition. The lost cause, indicated for each condition above the graph, is represented by the dashed line.

3 Experiment 2

To control for differences in the domains, this experiment used the same paradigm, but with three academic classes as the domains making all feedback on the same scale of grades. Classes were chosen so that there would be no clear *a priori* differences in domain importance. We predicted that participants in Experiment 2 should cut their losses in the lost cause, regardless of which domain was instantiated as the lost cause.

The same procedure from Experiment 1 was used with 72 participants. The three domains in this experiment were three academic classes, History, English, and Math. Participants were instructed to allocate their studying time among the three classes as they wanted over the course of 100 simulated days.

3.1 Results and Discussion

Mean importance ratings show that English was rated less important than History (t (71) = 4.95, p < .01) and Math (t (71) = 2.67, p < .01). There was no difference in importance ratings between History and Math (t (71) = 2.076, p = .05). Figure 2 shows the mean choice proportion for each domain across trials. In each condition, participants initially allocated more resources to the lost cause (presumably attempting to keep the three domains equal before realizing it was a lost cause), but decreased their choices for the lost cause as trials progressed. There was a significant interaction between change scores from each domain and the three lost cause conditions (F (4, 154) = 10.55, p. < .01, partial eta^2 = 2.15). The decrease in choices for the lost cause was significant for English (M = -1.42, t (32) = -3.729, p < .01), History (M = -1.33, t (26) = -2.66, p = .013), and Math (M= -1.95, t (20) = -4.34, p < .01).

Experiment 2 showed that when the three domains were comparable, participants cut their losses. That is, they behaved quite normatively. We designed the stimuli so that the three domains would be equally important. Though the importance ratings differed, this was driven primarily by English being rated less important than History or Math. The difference in ratings between History and Math was very small and non-significant with no class clearly more important than the other two. To further test the explanation that the importance of the domain is crucial to chasing a lost cause Experiment 3 was designed to extend both of the previous experiments using three academic classes but manipulating the importance of one.

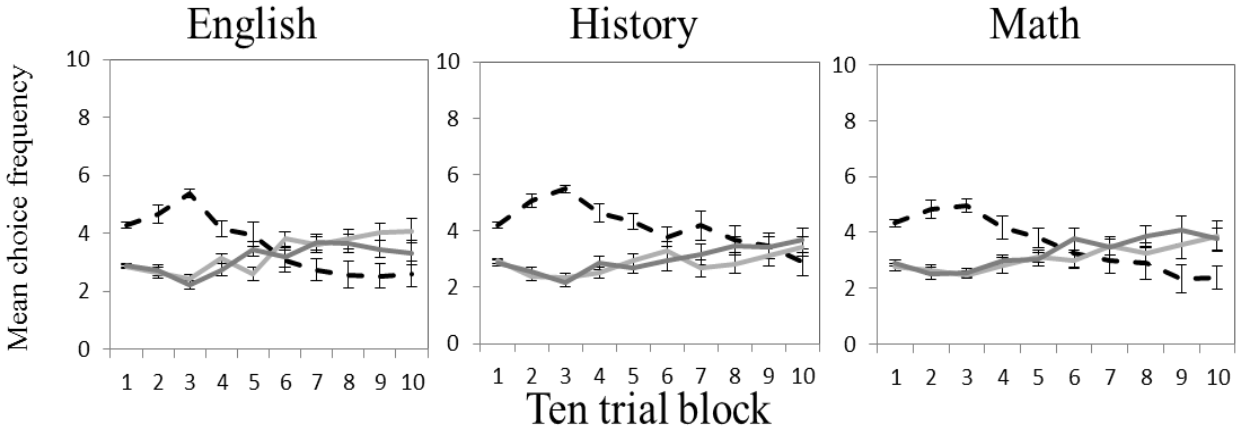


Figure 2. Experiment 2 choice frequency. The mean choice frequency across block of ten trials for each domain is displayed for each condition. The lost cause, indicated for each condition above the graph, is represented by the dashed line.

4 Experiment 3

In this experiment we used the academic classes of History, Sociology, and Psychology. The importance of one domain, Psychology, was manipulated by telling Participants that "although you would like to have the highest GPA at the end of the quarter, Psychology is required for your major and you need to obtain a C or better to avoid retaking the class".

Experiment 3 followed the same procedure as the first two experiments with 96 undergraduate participants with the exception of the additional instructions.

4.1 Results and Discussion

The manipulation of domain importance was successful with participants rating Psychology as more important than History (t (95) = 10.31, $p < .01$), and Sociology (t (95) = 9.06, $p < .01$).

Mean choice frequencies for are shown in Figure 3. In the conditions where Sociology and History were the lost cause, choices mirrored the results from Experiment 2. Testing choice frequency change scores against 0 show that participants cut their losses for both Sociology (M = -1.94, t (32) = -4.31, $p < .01$) and History (M = -1.94, t (32) = -5.016, $p < .01$). In the condition where the manipulated domain, Psychology, was the lost cause, participants chased the lost cause, increasing their allocation of time for Psychology (M = 1.4, t (32) = 2.986, $p < .01$).

The mean value of each domain (essentially their GPA at the end of the game) reflects the optimality of choice strategy. In the Sociology and History conditions, participants on average failed the lost cause, but passed their other two classes (the mean scores of 49 and 25 are represented as a C and D respectively during the game), while in the Psychology condition, where participants chased the lost cause, the mean ending values indicate that on average a participant in this condition spent so much time on Psychology that he or she failed all three classes. Averaging the final status of the three domains for an individual is analogous to a semester GPA. Using this measure participants in the Psychology condition were significantly lower than participants in both the History condition (t (61) = 5.49, $p < .01$, d = 1.4) and the Math condition (t (61) = 6.18, $p < .01$, d = 1.6). There was no difference between the History and Math conditions (t (64) = .393, p = .696, d = .09).

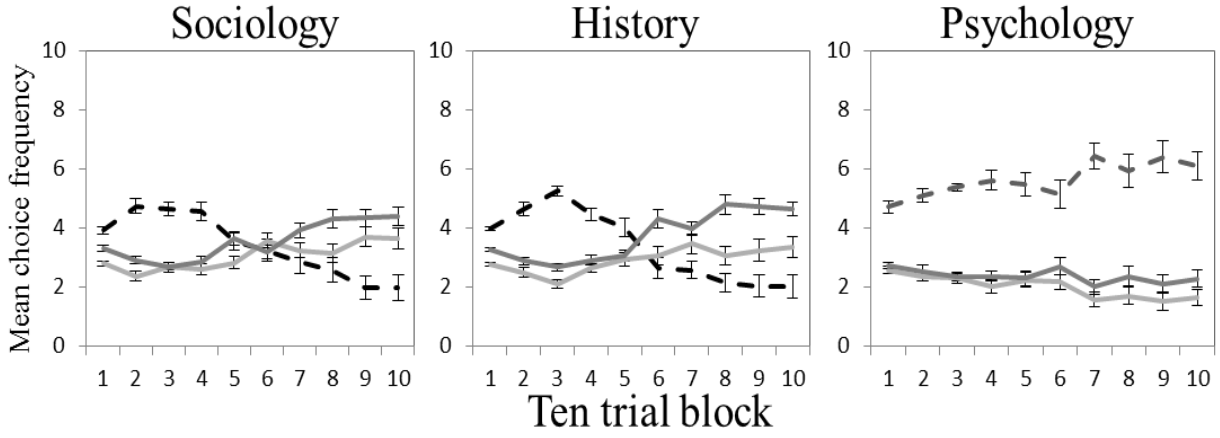


Figure 3. Experiment 3 choice frequencies. The mean choice frequency across block of ten trials for each domain is displayed for each condition. The lost cause, indicated for each condition above the graph, is represented by the dashed line.

5 Discussion

There is a large body of work in judgment and decision making detailing how human choices deviate from what are considered rational or normative standards (Shafir & LeBoeuf, 2002). Our work follows, expands, and qualifies this basic finding by showing that in repeated choices, people will chase a lost cause when the domain is important. In other situations, individuals are able to cut their losses. In contrast to sunk cost studies, the present tasks made the opportunity costs available but showing participants that choosing other options would improve the overall status of things, whereas remaining in the lost cause will eventually diminish all gains. This was sufficient information for individuals to make adjustments in domains that they cared equally about but not when one domain was more important.

Aging, negative life events, and economic forces are but a few factors that can place people in a position where they would be better off disengaging from an activity once enjoyed (Worsch, et al., 2004; Dohrenwend & Dohrenwen, 1974; Held, 1986; Wrosch & Freund, 2001). For example, new parents may find that they are unable to spend the same amount of time pursuing leisure activities as they once could. Beyond external changes, personal choices about time allocation can have detrimental effects on well being. Expanded work hours for example have negative effects on marital relations (White & Keith, 1990). Our results could lead to research on possible interventions designed to improve well being, including making opportunity costs more salient and reframing a problematic domain to minimize its perceived importance.

5 References

Arkes, H.R & Blumer, C. (1985). The psychology of sunk cost. *Organizational Behavior and Human Decision Processes*, 35, 124–140.

Dohrenwend, B. S., & Dohrenwend, B. P. (1974). *Stressful life events: Their nature and effects*. Oxford England: John Wiley & Sons. Held, 1986

Held, T. (1986). Institutionalization and deinstitutionalization of the life course. *Human Development*, *29*(3), 157-162.

Shin and Arieley (2004). "Keeping Doors Open: The Effect of Unavailability on Incentives to Keep Options Viable." *Management Science*, Vol. 50, No 5: 575-586.

Wrosch, C., & Freund, A. M. (2001). Self-regulation of normative and non-normative developmental challenges. *Human Development*, *44*(5), 264-283. doi:10.1159/000057066

Wrosch, C., Scheier, M. F., Miller, G. E., Schulz, R., & Carver, C. S. (2003). Adaptive Self-Regulation of Unattainable Goals: Goal Disengagement, Goal Reengagement, and Subjective Well-Being. *Personality and Social Psychology Bulletin*, *29*(12), 1494-1508. doi:10.1177/0146167203256921

White L, Keith B. The effect of shift work on the quality and stability of marital relations. Journal of Marriage and the Family. 1990;52:453 – 462.

The Moveable Feast of Predictive Reward Discounting in Humans

Luke Dickens
Brain & Behaviour Lab
Dept. of Computing
Imperial College London

Bernardo Caldas
Brain & Behaviour Lab
Dept. of Computing
Imperial College London

Benedikt Schoenhense
Brain & Behaviour Lab
Dept. of Bioengineering
Imperial College London

Guy-Bart Stan
Control Engineering Synthetic Biology Lab
Centre for Synthetic Biology and Innovation
Dept. of Bioengineering
Imperial College London

A. Aldo Faisal
Brain & Behaviour Lab
Dept. of Bioengineering
& Dept. of Computing
Imperial College London
& MRC Clinical Sciences Centre, London

Abstract

This work investigates the implicit discounting that humans use to compare rewards that may occur at different points in the future. We show that the way discounting is applied is not constant, but changes depending on context and in particular can be influenced by the apparent complexity of the environment. To investigate this, we conduct a series of neurophysics experiments, in which participants perform discrete-time, sequential, 2AC tasks with non-episodic characteristics and varying reward structure. The varying rewards in our games cause participants behaviour to change giving a characteristic signal of their future reward discounting. Model-free, model-based and hybrid reinforcement learning models are fit to participant data, as well as a lighter weight model which does not assume a learning mechanism. Results show that the complexity of the task affects the geometric discount factor, relating to the length of time that participants may wait for reward. This in turn indicates that participants may be optimising some hidden objective function that is not dependent on the discount factor.

Keywords: reinforcement learning, systems neuroscience, geometric-discount, planning horizon, human learning, human discount factor, psychophysics, task complexity

Acknowledgements

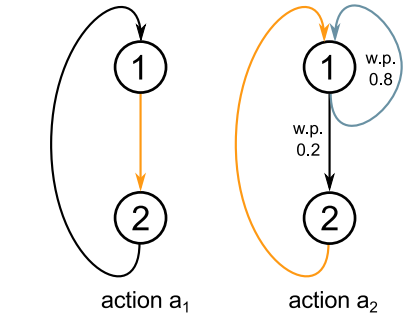
We are deeply indebted to EPSRC for their generous support of this research. This includes the Doctoral Prize Fellowship (Digital Economy) 2011/12, and project numbers $EP/M002187/1$ and $EP/G036004/1$. LDs present address: Dept. of Information Studies, University College London. Address for correspondence: aldo.faisal@imperial.ac.uk.

1 Introduction

Our brains adapt our behaviour in order to improve some measure of success in the world. Because our life is a continuous, ongoing experience, the mechanism by which we learn cannot always rely on having previously experienced present conditions, nor is it always possible to partition previous experience into well defined parts. In the terminology of reinforcement learning, we learn mostly *on-line* in a *non-episodic* environment, and we are (hopefully) *perpetual learners*, i.e. the learning is never completed.

In sequential decision making tasks, rewards may occur distributed over time, yet need to be related to the present time for decision making. To address this, a reward, r, that arises Δ units of time in the future, can be viewed as having the same value as an immediate reward $f(\Delta)r$, where $f(\Delta) \leq 1$ is a monotonically non-increasing function (i.e. f is a discount function). This is often viewed as a *trick* to allow learning (behavioural adaptation) to occur as soon as new experience is available. We call $f(\Delta)r$ the *present value* of future reward r. This discounting of future rewards may be more than just a mathematical convenience *trick*. For instance, it is often also used to explain how humans defer immediate gratification in favour of more substantial long-term outcomes, e.g. in neuroscience and economics. However, two agents that model the world in the same way, but use different discount functions f, will not have exactly the same preferences, nor will their decisions always coincide with the optimal choice evaluated over their lifetimes. As a consequence, a sophisticated on-line learning agent may sometimes wish to change their discount function f depending on the context, e.g. the complexity of the task, and we show evidence that humans do precisely that.

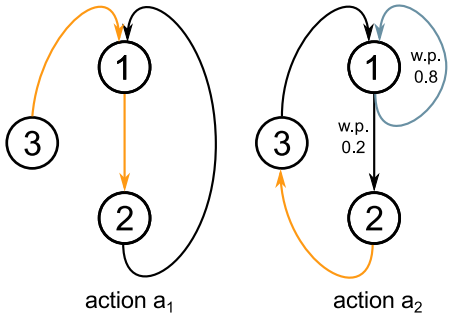
Game 1A	
$r(1, a_1, 2)$	\$0
$r(2, a_1, 1)$	\$25
$r(1, a_2, 1)$	\$25
$r(1, a_2, 2)$	\$0
$r(2, a_2, 1)$	\$x



Game 2A		
$r(1, a_1, 2)$	\$0	
$r(2, a_1, 1)$	\$50	w.p. x
	\$0	w.p. (1-x)
$r(1, a_2, 1)$	\$10	
$r(1, a_2, 2)$	\$0	
$r(2, a_2, 1)$	\$10	

Games type A

Game 1B	
$r(1, a_1, 2)$	\$0
$r(2, a_1, 1)$	\$5
$r(3, a_1, 1)$	\$x
$r(1, a_2, 1)$	\$20
$r(1, a_2, 2)$	\$0
$r(2, a_2, 3)$	\$0
$r(3, a_2, 1)$	\$25



Game 2B		
$r(1, a_1, 2)$	\$0	
$r(2, a_1, 1)$	\$10	
$r(3, a_1, 1)$	\$80	w.p. x
	\$0	w.p. (1-x)
$r(1, a_2, 1)$	\$10	
$r(1, a_2, 2)$	\$0	
$r(2, a_2, 3)$	\$0	
$r(3, a_2, 1)$	\$10	

Games type B

Figure 1: Transition dynamics (central) games played by participants in experiments, step-wise effects of the two actions, a_1 and a_2, shown separately for clarity. Reward structures (left/right) describe two games for each game type – $r(s, a, s')$ is reward for transitioning from state s to s' under action a. Non-deterministic binary outcomes are labelled w.p. (with probability). All games have non-deterministic transitions in state 1 for action a_2. Games 2A & 2B have some non-deterministic rewards. The value of x changes slowly during play.

Related work Reinforcement learning models offer a biologically plausible framework in which to study human behaviour in sequential learning tasks [3–5]. In particular, evidence for reward prediction errors in the brain, have a close analog to the temporal differences in the widely used temporal difference (TD) learning algorithm [4].

In discrete time tasks, some RL methods, including TD, use a geometrically discounted reward, often called $\gamma \in [0, 1]$. Here, a guaranteed reward, r, lying k time-steps in the future is assigned present value $\gamma^k r$. Values of $\gamma < 1$ allow on-line algorithms (those that continuously incorporate new experience) to *approximately* optimise the average reward per time-step [6]. However, for machine learning practitioner γ is typically a free parameter, which is chosen a priori to

suit the learning conditions. The choice of γ can have a profound effect on the trade-off between accuracy and speed of convergence [7], with larger γ leading to more accurate but slower learning.

The work from [1] tests human participants on episodic state-based 2AC choice tasks, and fits multiple RL models to the data, including model-based[1], model-free, and hybrid (combined model-based/model-free, see [2]) variants. They find that a participants' behaviour is best described by hybrid models, and estimate γ values close to 1. However, they use tasks with episodic structure, where the length of episodes is fixed. In these tasks, it is sufficient for participants to optimise the average reward per episode, and update learning after each episode, therefore avoiding the need for on-line learning .

Tasks with non-uniform length to their episodes (and unknown overall time constraints), require participants to optimise over a longer, uncertain period of time, and it is no longer feasible to use $\gamma = 1$. Work described in [5] investigates such a task, where they find that participants with chemically elevated seratonin levels appear to use lower γ values, than when under control conditions. The authors link lower γ values to more impulsive behaviours, i.e. geared towards more immediate gratification. This in turn, may help to explain some of the characteristic behavioural changes that are associated with recreational drug use. However, the work from [5] only fit one RL model to the data, without the same comparative evaluation across multiple models seen here and in [1]. More importantly, we investigate how different degrees of task complexity affect the induced reward discounting.

Contributions The proposed poster will demonstrate the following contributions:

- We develop a suite of discrete-time sequential state-based 2AC (two action choice) tasks, that induce behaviours designed to elicit human reward discounting characteristics.
- We consider a suite of model-free, model-based, and hybrid reinforcment learning (RL) methods, and infer the maximum-likelihood parameters for each model on each participant's data, including the geometric discount γ. We call these *bottom-up* analysis methods.
- We develop a (*top-down*) method for inferring γ without assuming a specific learning mechanism.
- We compare both approaches on synthetic data, and show that bottom-up methods (unsurprisingly) perform more accurately on data generated by a matching model. Conversely, our top method accurately recovers γs from data generated by a wide range of RL methods and parameters.
- Some model-free RL methods consistently outperform other RL models on experimental data. The top-down method consistently gives predictions of γ, which are more stable than the best bottom-up methods and have greater evidential weight (using Akaike's information criterion (AIC)).
- More complex tasks are associated with higher values of γ (using both bottom-up and top-down analysis), and γ has a strong relationship with the characteristic time-scale of the task. However, noise characteristics have at best a weak effect on the induced value of γ.

This last finding indicates that an individual's discounting function can change depending on context, and appears to be influenced by the characteristic time-scale of the task. This may be surprising to some RL researchers, as the vast majority of RL algorithms that use a discounting function keep that function fixed.

2 General Approach

We design sequential state-based 2AC tasks, whose reward structure change slowly, but unpredictably, during play at a medium time-scale. Participants are told to optimise the total reward over the duration of a game, and the changing reward structure forces them to continuously explore and adapt behaviour at the short time-scale to achieve this.

We define a participant's *planning timescale* as the effective number of time-steps in the future, k, that a reward must be for its *present value* to half that of its *immediate value*, i.e. $\frac{r}{2} = f(k)r$. The task is designed such that a user is repeatedly presented with a choice between an immediate small reward versus a longer term, larger reward. We refer to this as a choice between shorter and longer paths. The changes in the reward structure are such that there are periods of the experiment where even participants with very short planning timescale would still prefer to wait for larger rewards. Conversely, there are also periods where even participants with very long planning timescales prefer the short term rewards.

These changes in reward structure will therefore induce behavioural changes in the participants, causing them to switch between preferring shorter paths to preferring the longer paths, and vice versa. These switches can help us to identify each individual's planning timescale, l, in terms of a step-wise geometric discount γ (where $l = \frac{\ln 0.5}{\ln \gamma}$). A variety of games

[1] As with [1], we differentiate between model-free RL methods (pure value based learning), and model-based methods (e.g. including an environmental model to accelerate learning and improve predictions).

are used with different levels of complexity. Some games present a choice between waiting 1 or waiting 2 time-steps for a reward. Other games present a choice between waiting 1 or waiting 3 time-steps for a rewards. We also explore differences between games with deterministic but changing game rewards (Games 1A & 1B, see Section 4), versus games with probabilistic rewards with changing distributions (Games 2A & 2B).

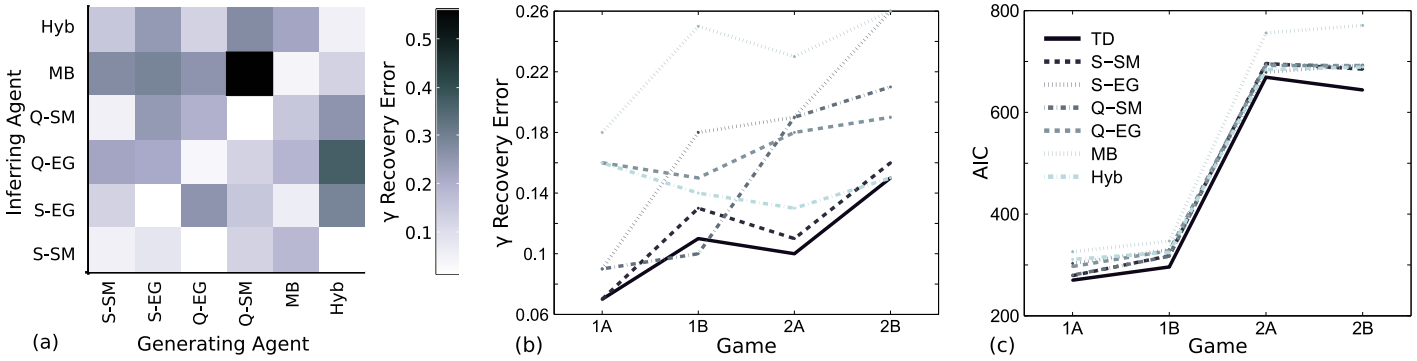


Figure 2: Performance on synthetic data (a) Average γ-recovery accuracy, $||\hat{\gamma} - \gamma||_2$. Data generated on game 1A using *generating agent* and a selection of parameters, then γ recovered by inferring agent. (b) Average γ-recovery accuracy on synthetic data averaged over all generating agents for a variety of generating parameters. (c) Corresponding averaged Akaike's information criterion (AIC). The best performance for each game is highlighted in bold.

3 Results

It is unclear a priori what learning approach participants will employ, whether or not it is equivalent to an RL method, and if so whether it is model-based, model-free or hybrid. We use a gradient based method to find the maximum-likelihood parameters for each method from observed traces of states, actions and rewards. A top-down method is also developed, which assumes that a geometric factor is used to discount future rewards, but without assuming a specific learning mechanism. For brevity the following shorthand is used for inference methods and RL algorithms: Top-down (TD), SARSA Softmax (S-SM), SARSA ε-greedy (S-EG), Q-Learning Softmax (Q-SM), Q-Learning ε-greedy (Q-EG), Model-based (MB) & Hybrid method combining Q-EG + MB (Hyb).

Synthetic Data Figure 2 (a) shows the root mean squared (RMS) γ-recovery accuracy between the recovered discount factor, $\hat{\gamma}$, and the generating value, γ, on synthetic data for a variety of methods. Each RL methods in the suite is used both to generate synthetic data, and to recover γ. As expected, the recovery performance is best when the inferring agent is the same as the generating agent. Figures 2 (b) and (c) respectively show the average RMS recovery accuracy and the average AIC of each bottom-up and the top-down method on synthetic data. Data is averaged over a selection of all agents, each with a variety of the parameters. The top-down method performs the best overall, and competes with the best recovering model in all cases. Also, the top-down method has the best average AIC[2] of all methods.

Experimental Data We next apply these methods to the experimental participant data. We fit each bottom-up and the top-down models to each participant's behaviour, and evaluate the strength of evidence for the given model (using AIC). On the majority of individual participants, and averaged overall, the top-down model has the lowest AIC of all models. Figure 3 (a) shows the mean recovered discount factors $\hat{\gamma}$ for each game using both the top-down (TD) method and the two best performing RL methods. We find $\hat{\gamma}$ in reasonable agreement between these algorithms, both at the population level and individually. To our knowledge, this is the first direct evidence that humans change their discounting function depending on the complexity of the associated game, where complexity is measured in terms of characteristic timescale between rewards. This shows an average $\hat{\gamma} \in [0.55, 0.6]$ for games with a long-path of length 2 (1A & 2A) and an average $\hat{\gamma} \in [0.75, 0.9]$ for games with a long path of length 3 (1B & 2B). There is no discernable difference between games with deterministic and non-deterministic rewards, e.g. Game 1A versus Game 2A.

To explore these results further, we plot in Figure 3 (b) the individually inferred discount factors $\hat{\gamma}$ against the average experienced path length, $\bar{l}$ (time between visits to state 1). The dotted line represents the orthogonal least squares fit to the data after individual variances have been normalised, and the positive slope is in agreement with our hypothesis that longer average pathlengths lead to longer horizons. The pair of variables $\hat{\gamma}$ and $\bar{l}$ have a Pearson's product-moment correlation coefficient of $r = 0.1833$, and the corresponding 1-tailed test for a significant positive linear relationship

[2]AIC is chosen over BIC for simplicity. However, BIC penalises complexity, in terms of number of parameters, more heavily. RL methods have more parameters than the top-down method, so the top-down approach would be preferred by either measure.

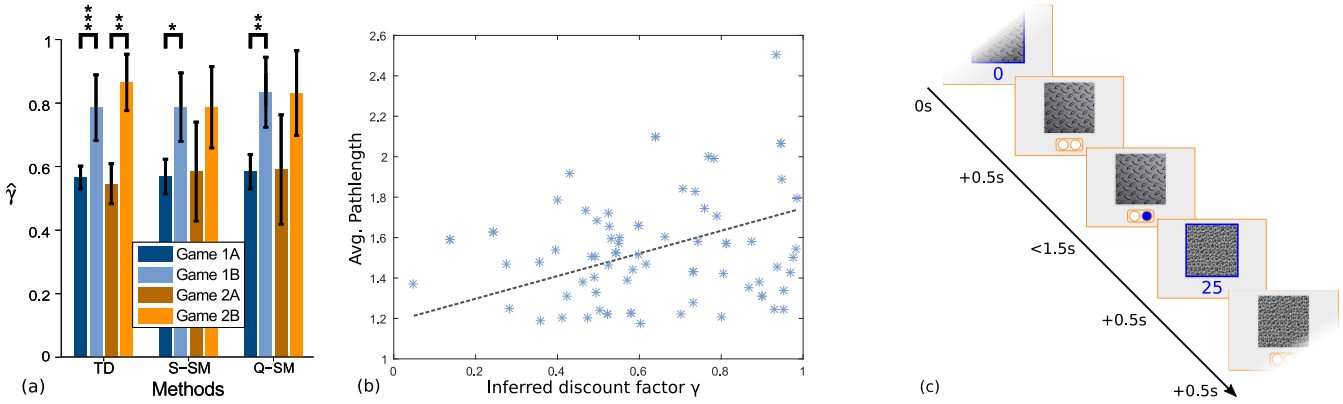


Figure 3: (a) Average inferred discount factors $\hat{\gamma}$ on per game basis, with top-down and two best performing RL methods shown. (b) Individually inferred discount factors $\hat{\gamma}$ versus the average pathlength, $\bar{l}$, experienced by each participant in each game. The dotted line shows the orthogonal least squares fit after normalising the variance. (c) Timeline of events in a single step of a task. The current state is presented (phase 0), a choice is requested (phase 1), the choice is then made – or assigned randomly on time-out (phase 2), the next state and reward is displayed (phase 3/0), and the game continues.

between the two variables gives a p-value of 0.055. The AIC values on participant data give greater support for model-free reinforcement learning models than the model-based or hybrid models (not shown). This is in contrast to the findings of [1] on fixed length episodic tasks. However, further investigations are required to determine if this is significant.

4 Methodology

Psychophysics experiments (N=24 participants) involved basic computer interaction and were conducted in accordance with local ethics committee guidelines. Participants were presented with sequential decision making tasks (see Figure 2 (c)). At each time-step, the participant is presented with an image representing the current state. After a pause of 0.5 seconds, the participant chooses one of two actions within a further 1.5 seconds (or a random choice is made and displayed). After another 0.5 seconds, the resulting state is presented along with a numeric reward, indicating the value of the most recent transition. The reward structure varies slowly so participants must continually explore and adapt. Participants are not told how long the task will last, and therefore must view the task as one with an unknown horizon. There were 4 fundamental tasks, shown in Figure 1, and each participant was presented all 4 tasks in a random order with breaks. Each task (game) repeatedly presents the participant with (limited) control over whether she takes a longer path leading to a larger reward, or a shorter path that leads to a smaller reward. The expected value of the larger reward is varied throughout play. The shorter path in each game is of length 1 time-step. Games of type A (Game 1A & 2A) have a long path length of 2 steps. Games of type B (Game 1B & 2B) have a long path length of 3 steps. Participants are instructed to try to achieve as large a total reward as possible over the lifetime of each game.

Top Down Method To predict γ with the top-down method, we (a) estimate the underlying policy at each time-step, (b) identify the time points in the trace when the dominant action in the policy changes, and (c) use gradient ascent to determine the maximum likelihood value for the gaussian/exponential filter width over recent rewards and a corresponding γ which best explains the switching points.

References

[1] N. Daw, S.J. Gershman, B. Seymour, P. Dayan, and R.J. Dolan. Model-Based Influences on Humans' Choices and Striatal Prediction Errors. *Neuron*, 69(6):1204–1215, 2011.

[2] J. Gläscher, N. Daw, P. Dayan, and J.P. O'Doherty. States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron.*, 66(4):585–95, 2010.

[3] A. R. Otto, A. Skatova, S. Madlon-Kay, and N. D Daw. Cognitive Control Predicts Use of Model-based Reinforcement Learning. *Journal of Cognitive Neuroscience*, 27(2):319–333, 2015.

[4] W. Schultz, P. Dayan, and PR. Montague. A Neural Substrate of Prediction and Reward. *Science*, 275(5306):1593–9, 1997.

[5] N. Schweighofer, M. Bertin, K. Shishida, Y. Okamoto, S.C. Tanaka, S. Yamawaki, and K. Doya. Low-serotonin levels increase delayed reward discounting in humans. *The Journal of Neuroscience*, 28(17):4528–4532, 2008.

[6] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[7] Huizhen Yu and D.P. Bertsekas. New error bounds for approximations from projected linear equations. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 1116–1123, Sept 2008.

KWIK Inverse Reinforcement Learning

Vaishnavh Nagarajan
Indian Institute of Technology Madras
vaish@cse.iitm.ac.in

Balaraman Ravindran
Indian Institute of Technology, Madras
ravi@cse.iitm.ac.in

Abstract

Imitation learning or learning from demonstrations is a means of transferring knowledge from a teacher to a learner, that has led to state-of-the-art performances in many practical domains such as autonomous navigation and robotic control. The aim of the learning agent is to learn the expert's policy through trajectories demonstrated by the expert. One solution to this problem is inverse reinforcement learning (IRL), where the learner infers a reward function over the states of the Markov Decision Process on which the mentor's demonstrations seem optimal. However, since expert trajectories are practically expensive, it becomes crucial to minimize the number of trajectory samples required to imitate accurately. Moreover, when the state space is large, the agent must be able to generalize knowledge acquired from demonstrations covering a small subset of the state space, confidently to the rest of the states. To address these requirements, we first propose a novel reduction of IRL to classification where determining the separating hyperplane becomes equivalent to learning the reward function itself. Further, we also use the power of this equivalence to propose a Knows-What-It-Knows (KWIK) algorithm for imitation learning via IRL. To this end, we also present a novel definition of admissible KWIK classification algorithms which suit our goal. The study of IRL in the KWIK framework is of significant practical relevance primarily due to the reduction of burden on the teacher: a) A self-aware learner enables us to avoid making redundant queries and cleverly reduce the sample complexity. b) The onus is now on the learner (and no longer on the teacher) to proactively seek expert assistance and make sure that no undesirable/sub-optimal action is taken.

Keywords: Reinforcement Learning; Markov Decision Process; Learning from Demonstrations; Imitation Learning; KWIK; Knows what it knows; Online learning; Inverse Reinforcement Learning;

1 Introduction

Imitation learning is broadly solved in two different ways. One approach is to pose it as a supervised learning problem where a classifier learns the action labels for all states in the state space based on training data (the expert's demonstrations). The other approach is a model-based solution that uses inverse reinforcement learning (IRL) to find a mapping from states to real-valued rewards that makes the expert trajectories seem optimal. The learner hence follows the optimal policy on these rewards. IRL methods have the advantage of representing the acquired knowledge succinctly as rewards over the states.

In practice, both these approaches could suffer from a considerable burden on the teacher who is expected to produce sufficient trajectories for accurate imitation. What makes this more undesirable is that many of the trajectories happen to be redundant and yet expensive. To this end, Judah et al. [4] have proposed active learning algorithms for supervised learning based imitation learning and analyse their PAC label complexity. On the other hand, there has been very little work on formally understanding active imitation learning through IRL. Silver et al. [8] have studied active learning heuristics where the learner requests trajectories in such a way that the knowledge about the reward function acquired is either novel or reduces uncertainty in the current beliefs. Lopes et al. [6] only provide an empirical technique to actively query by choosing states with the greatest uncertainty with respect to the policy that was learnt using Bayesian IRL.

A drawback with all of the above active learning algorithms is that they assume that the learner has complete access to the state space and can also query the expert for a demonstration on any of these states. Often this might not be desirable because some states may not even be realizable and furthermore, this knowledge might not be accessible to the learner. Chernova and Veloso [2] address this by allowing the learner to 'interactively' request the teacher's demonstrations whenever it encounters a state where its confidence on the learnt policy is below a threshold. They provide an algorithm that pertains only to the supervised learning based imitation learning and support it with empirical results.

To overcome these multiple issues, we propose the novel idea of considering imitation learning in the Knows-What-It-Knows (KWIK) framework [5]. A KWIK algorithm is an online learning algorithm that is considered to be self-aware i.e., if and only if the learner believes that it has insufficient experience to predict on a new sample, does the learner ask the expert for the answer. Considering imitation learning in this framework significantly benefits us in four ways. First, we overcome the problems that come with allowing the learner to query on any arbitrary state. The learner makes queries only on the states that it encounters. Secondly, we are able to allow the learner to enact its policy and learn on-the-fly. Thirdly, the burden on the expert is substantially reduced as the learner only selectively requests demonstrations. Finally, we are guaranteed that the learner does not mistakenly assume that it knows what to do when it actually does not. This is of practical importance because we would not want the learner to take a non-optimal and possibly dangerous action which could have been avoided if the expert had intervened.

Though Walsh et al. [10] study what is called as a generalized apprenticeship learning protocol in relation to KWIK learnable classes, their problem domain, as they claim, is fundamentally different from the imitation learning problem that we consider. They study a learner that has access to the rewards during exploration, while the teacher augments this knowledge.

Next, we propose a reduction of the KWIK apprenticeship learning problem via IRL to KWIK classification. Note that this reduction to classification is not the same as direct imitation learning methods that use a classifier to learn a mapping from states to actions. We are primarily interested in finding the unknown reward function defined over the state-action pairs and not just what action label a state corresponds to. We show that learning the reward function is equivalent to learning the separating hyperplane in the classification problem.

Our next contribution in this work is a novel definition of KWIK classification that applies to this equivalence in imitation learning. The KWIK framework requires that the learner achieves point-wise accuracy, unlike in a PAC-learner. That is, if the learner makes a prediction on a new sample without seeking expert advice, the learner must be ϵ-accurate. However, it is not possible to define ϵ-accuracy on discrete labels (unless we consider a continuous action space in which case we would opt for KWIK online regression algorithms [9]). One could overcome this by defining accuracy with respect to the prediction about the distance of the sample from the separating hyperplane, as considered in some *selective sampling* algorithms [1, 3]. However, these algorithms have significantly different assumptions than that expected by the KWIK imitation learning agent and are hence not applicable. We further motivate the validity of our definition of KWIK classification by providing polynomial KWIK bounds for 1-D classification. Finally, we also provide a KWIK protocol for imitation learning that uses an underlying KWIK classifier that suits our requirement that the learner takes ϵ-optimal actions.

2 Preliminaries

Definition A Markov Decision Process (MDP) is represented as a 5-tuple (S, A, T, γ, R) where S is a set of states; A is a set of actions; T is a set of state transition probabilities; $\gamma \in (0, 1]$ is a discount factor; and $R : S \rightarrow A$ is the reward function.

We assume that the state-action pairs can be mapped to a $k-$dimensional vector of features, $\phi : S \times A \rightarrow [0,1]^k$. For example, in a maze where a puddle has to be avoided we could have three boolean features each describing whether the state is a puddle or a goal or neither. Thus, the actual reward $R(s,a)$ corresponding to an action at a state is equal to $\vec{w} \cdot \phi(s,a)$ where $w \in \mathbb{R}^k$. We need to ensure that $||w||_1 \leq 1$ so that the rewards themselves are upper-bounded by 1.

A policy π is a mapping from states to (probability distributions over) actions. The value of a state-action pair under a policy π is

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi[R(s,a) + \sum_{t=1}^{\infty} \gamma^t R(s_t,a_t)|\pi] \\ &= \vec{w} \cdot \mathbb{E}_\pi[\phi(s,a) + \sum_{t=0}^{\infty} \gamma^t \phi(s_t,a_t)|\pi] \end{aligned}$$

where $s_1, s_2 \ldots$ are the subsequent states visited by following the policy and $a_1, a_2 \ldots$ are the corresponding actions taken at those states. Thus we see that the $Q-$values can also be linearly parametrized. We assume that the learning agent is presented these $Q-$value parameters in the form of $\phi_Q : S \times A \rightarrow \mathbb{R}^k$. The IRL task now reduces to interacting with the expert and making an accurate estimate of the vector $\vec{w}$.

Definition We define the **Knows-What-It-Knows protocol** with parameters (ϵ, δ) ($\epsilon \in [0,1], \delta \in [0,1)$) as follows . Consider a hypothesis space $H \subseteq (X \rightarrow Y)$ from which the adversary picks a target function $h^* \in H$. During a run, for each time-step the adversary picks an input $x \in X$ for which the learner either emits a $\perp$ (dont-know) or makes a prediction $\hat{h}(x)$. If the learner emits $\perp$, the adversary informs the learner of $h^*(x)$. For an algorithm to be an admissible KWIK algorithm, we need that with probability $1-\delta$ the following conditions hold good:

- Whenever the algorithm makes a prediction $\hat{h}(x)$, $|\hat{h}(x) - h^*(x)| \leq \epsilon$
- The number of time-steps for which the algorithm emits $\perp$ is bounded by $B(\epsilon, \delta)$ a function that is polynomial in $1/\epsilon$, $1/\delta$ and some parameters that define H.

3 KWIK-Learner for Binary Classification

Definition We define **an admissible KWIK-learner for binary classification** as follows. We assume that the hypothesis class is the set of separating hyperplanes $\{\vec{w} | \vec{w} \in \mathbb{R}^k, ||\vec{w}||_1 \leq 1\}$. If the adversary picks a $\vec{w}^*$, the correct label of $\vec{x}$ is given by $\text{SGN}(\vec{w}^* \cdot \vec{x}) \in \{+1, -1\}$. For the learner to be admissible, the following must hold good for every run, with probability $1-\delta$:

- Whenever the algorithm makes a prediction on x, if $|\vec{w}^* \cdot \vec{x}| > \epsilon$, then $\hat{h}(\vec{x}) = \text{SGN}(\vec{w}^* \cdot \vec{x})$
- The number of time-steps for which the algorithm emits $\perp$ is bounded by $B(\epsilon, \delta)$ a function that is polynomial in $1/\epsilon$, $1/\delta$ and k.

Intuitively, we require that the algorithm predicts correctly on all the points that are sufficiently far away from the separating hyperplane; if the sample point is within the $\epsilon-$margin of the hyperplane, we allow the classifier to make mistakes.

This may be compared to the KWIK-MB model proposed by Sayedi et al. [7] where the KWIK algorithm is also allowed to make a fixed number of mistakes. However, our model is significantly different in that we allow infinitely many mistakes but restrict them to a very small space around the separating hyperplane. If we did not allow the learner to make infinitely many mistakes, we would expect the learner to perennially refine its knowledge in the small space around the hyperplane. This might require exponentially many queries to accurately place the hyperplane. Furthermore, we will see that our condition also eventually suits our imitation learning problem where the learner is required to be ϵ-optimal.

Next, we discuss assumptions about noise in the expert's labels. In the KWIK framework, we assume that the noisy observation produced by the expert has an expectation equal to the correct output. Thus, for classification we assume a teacher to be (ϵ_T, ϵ_Y)-optimal, if the teacher outputs y for an input x such that:

$$\begin{aligned} \mathbb{E}[y] &> \epsilon_Y \quad \text{if } \vec{w}^* \cdot \vec{x} \geq \epsilon_T \\ \mathbb{E}[y] &< -\epsilon_Y \quad \text{if } \vec{w}^* \cdot \vec{x} \leq -\epsilon_T \end{aligned}$$

In other words, we expect that for all input points that are at least ϵ_T away from the separating hyperplane, the expert labels them correctly with probability at least $1/2 + \epsilon_Y/2$. A good teacher will have a high ϵ_Y and a small ϵ_T. We note that this is a significantly relaxed assumption when compared to the selective sampling approach of Dekel et al. [3] where they assume that the accuracy of the expert increases with the distance from the separating hyperplane.

3.1 A simple KWIK 1-D classification algorithm

We analyse a naive algorithm for 1-D classification to demonstrate how the KWIK conditions we proposed allow us to design algorithms with a KWIK-bound polynomial in $1/\epsilon$ and $1/\delta$ even under the relaxed noise assumption of the teacher's outputs. Assume the input space spans unit length and that $4\epsilon_T < 2\epsilon < \epsilon_Y$, which is natural because it is not possible for the learner to outdo the teacher.

The learning algorithm discretizes the input space into $\frac{2}{\epsilon}$ segments. When the adversary presents a sample belonging to a segment, the algorithm emits $\perp$ when the number of samples already queried in this segment is fewer than $\mathcal{O}(\frac{1}{\epsilon^2}\ln(\frac{2}{\epsilon\delta}))$. When the number of samples is however greater than this, we can show that if the segment is completely outside the ϵ_T-margin around the separating point, the proportion of queried points in this segment that would have been labelled correctly by the expert will be at least $1/2 + \epsilon_Y/2 - \epsilon > 1/2$ with a high probability of $1 - \epsilon\delta/2$. Thus, after acquiring sufficient samples in each of the segments, we will correctly learn the labels of all the segments outside an $\epsilon-$ margin of the separating point with probability $1 - \delta$. Thus, the number of queries made will be $\mathcal{O}(\frac{1}{\epsilon^3}\ln(\frac{2}{\epsilon\delta}))$.

4 KWIK Inverse Reinforcement Learning Protocol

We now present the reduction of IRL to KWIK classification. At any state $s \in S$, the learner is presented with a set of at most $|A|$ actions of which the learner is required to pick an $\epsilon-$optimal action. Let $\vec{w}$ be the unknown weight vector for the rewards. We assume that the learner has access to a KWIK classification algorithm whose input space is $\mathbb{R}^k$ and whose accuracy parameter is set to be $\epsilon/(|A| - 1)$.

For some $a^*, a' \in A$, we expect the classifier to predict $\text{SGN}(\vec{w} \cdot (\phi_Q(s, a^*) - \phi_Q(s, a')))$ given $(\phi_Q(s, a^*) - \phi_Q(s, a'))$ as input. If it predicts, say, $+1$, from the conditions we stipulated, we know that $\vec{w} \cdot (\phi_Q(s, a^*) - \phi_Q(s, a')) > \epsilon/(|A| - 1)$. We will use this property to use the classifier to identify the action that corresponds to nearly the highest Q-value.

If the classifier is unable to predict, and instead outputs a $\perp$ we request expert advice in the form of a preference over these pair of actions. We note that we could study various other modifications of this algorithm, where the expert only provides knowledge about the best action amongst all actions instead of pairwise preferences.

Algorithm 1 KWIK Inverse Reinforcement Learning Protocol

Require: Teacher $\mathcal{T}(\epsilon_T, \epsilon_Y)$ with true weight vector for rewards $\vec{w}$, Admissible KWIK Classifier $\mathcal{C}(\frac{\epsilon}{|A|-1}, \delta)$ with weight estimate for rewards $\hat{\vec{w}}$
for $t = 1, 2, \ldots$ **do**
  $s =$ Current State of the Environment
  $\hat{a}_{best} = a_1$
  for i = $2, \ldots |A|$ **do**
    Present $\phi_Q(s, a_i) - \phi_Q(s, \hat{a}_{best})$ to $\mathcal{C}$
    if Output of $\mathcal{C} = \perp$ **then**
      Present $\phi_Q(s, a_i) - \phi_Q(s, \hat{a}_{best})$ to $\mathcal{T}$
      $\mathcal{T}$ outputs noisy observation of $\text{SGN}(\vec{w} \cdot (\phi_Q(s, a_i) - \phi_Q(s, \hat{a}_{best})))$
      $\mathcal{C}$ learns from output of $\mathcal{T}$ and updates $\hat{\vec{w}}$
      if Output of $\mathcal{T} = +1$ **then**
        $\hat{a}_{best} = a_i$
    else
      $\mathcal{C}$ outputs $\text{SGN}(\hat{\vec{w}} \cdot (\phi_Q(s, a_i) - \phi_Q(s, \hat{a}_{best})))$
      if Output of $\mathcal{C} = +1$ **then**
        $\hat{a}_{best} = a_i$
  Perform $\hat{a}_{best}$

In Algorithm 1, at any state the learner scans all the possible actions (which is at most $|A|$) and maintains a candidate action that it considers to be the best amongst the actions that have been iterated through. The correctness of this algorithm would follow if we show that after iterating over all the actions, if the algorithm has not queried the teacher, it always chooses an ϵ-optimal action. We prove the following lemma from which the above statement follows directly by setting $i = |A|$.

Lemma 4.1 *After iterating over the first i actions, if the algorithm has not made any queries, the candidate action picked by the algorithm is $(i-1)\frac{\epsilon}{|A|-1}-$ optimal with respect to the best action amongst the first i actions.*

Proof The claim clearly holds good when $i = 1$. For any arbitrary round $i < |A|$ assume that the claim is true. That is, if $\hat{a}_i$ is the candidate action picked by the algorithm, and a_i^* is the best action amongst the first i actions, then:

$$\vec{w} \cdot \phi_Q(s, \hat{a}_i) \geq \vec{w} \cdot \phi_Q(s, a_i^*) - (i-1)\frac{\epsilon}{|A|-1} \tag{1}$$

If the algorithm chose a_{i+1} over $\hat{a}_i$, we know from the KWIK classifier conditions that

$$\vec{w} \cdot \phi_Q(s, a_{i+1}) \geq \vec{w} \cdot \phi_Q(s, \hat{a}_i) - \frac{\epsilon}{|A|-1} \tag{2}$$

If this decision were to be inconsistent with our claim, a_{i+1} must not be an $i\frac{\epsilon}{|A|-1}$-optimal action (among the $i+1$ actions). Then a_i^* must still be the best action amongst the first $i+1$ actions. However, from inequalities (1) and (2), we can see that:

$$\vec{w} \cdot \phi_Q(s, a_{i+1}) \geq \vec{w} \cdot \phi_Q(s, a_i^*) - i\frac{\epsilon}{|A|-1}$$

which makes a_{i+1} an $i\frac{\epsilon}{|A|-1}$-optimal action amongst the first $i+1$ actions, which is a contradiction.

On the other hand, if the algorithm still chose $\hat{a}_i$ over a_{i+1}, it would be inconsistent with our claim only if a_{i+1} was the best action amongst the $i+1$ actions and if $\hat{a}_i$ was not sufficiently optimal. That is,

$$\vec{w} \cdot \phi_Q(s, \hat{a}_i) < \vec{w} \cdot \phi_Q(s, a_{i+1}) - i\frac{\epsilon}{|A|-1} \tag{3}$$

However, this implies a much weaker inequality:

$$\vec{w} \cdot \phi_Q(s, \hat{a}_i) < \vec{w} \cdot \phi_Q(s, a_{i+1}) - \frac{\epsilon}{|A|-1}$$

which would have ensured that the KWIK classifier indicated that a_{i+1} was a better action. Hence, by induction we show that the lemma holds good for all $i = 1, \ldots |A|$. ∎

5 Conclusion and Future Work

In this work, we have provided an understanding of IRL in the KWIK framework, which has not been studied before. Apart from the practical relevance of a KWIK imitation learning algorithm, this setup provides scope for theoretical guarantees that can be provided for active query based IRL which have not been provided so far. We have also laid the groundwork for a new class of algorithms that can be termed as 'KWIK classifiers' that will suit tasks similar to IRL-based imitation learning. Since the existing selective sampling based classification approaches do not apply here due to their strong assumptions, an appropriate direction for future work will be to design admissible KWIK classification algorithms that suit our conditions.

References

[1] Nicolò Cesa-Bianchi, Claudio Gentile, and Francesco Orabona. Robust bounds for classification via selective sampling. In *ICML 2009*, volume 382, pages 121–128.

[2] Sonia Chernova and Manuela M. Veloso. Interactive policy learning through confidence-based autonomy. *CoRR*, abs/1401.3439, 2014.

[3] Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Robust selective sampling from single and multiple teachers. In *COLT 2010*, pages 346–358.

[4] Kshitij Judah, Alan Fern, and Thomas G. Dietterich. Active imitation learning via reduction to I.I.D. active learning. In *UAI 2012*, pages 428–437.

[5] Lihong Li, Michael L. Littman, Thomas J. Walsh, and Alexander L. Strehl. Knows what it knows: a framework for self-aware learning. *Machine Learning*, 82(3):399–443, 2011.

[6] Manuel Lopes, Francisco S. Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *ECML PKDD 2009 Part II*, pages 31–46.

[7] Amin Sayedi, Morteza Zadimoghaddam, and Avrim Blum. Trading off mistakes and dont-know predictions. In *NIPS 2010*, pages 2092–2100.

[8] David Silver, J. Andrew Bagnell, and Anthony Stentz. Active learning from demonstration for robust autonomous navigation. In *In IEEE ICRA*, pages 200–207.

[9] Alexander L. Strehl and Michael L. Littman. Online linear regression and its application to model-based reinforcement learning. In *NIPS 2007*, pages 1417–1424.

[10] Thomas J. Walsh, Kaushik Subramanian, Michael L. Littman, and Carlos Diuk. Generalizing apprenticeship learning across hypothesis classes. In *ICML 2010*, pages 1119–1126.

Hierarchical Decision Making using Spatio-Temporal Abstractions In Reinforcement Learning

Peeyush Kumar
University of Washington Seattle
agaron@uw.edu

Nikhil Nainani*
Indian Institute of Technology, Madras
nikhil.m.nainani@gmail.com

Ramnandan Krishnamurthy*
Indian Institute of Technology, Madras
nandparikrish@gmail.com

Balaraman Ravindran
Indian Institute of Technology, Madras
ravi@cse.iitm.ac.in

Abstract

This paper introduces a general principle for automated skill acquisition based on the interaction of a reinforcement agent with its environment. Our approach involves identifying a hierarchical description of the given task in terms of abstract states and extended actions between abstract states. Identifying such useful structures present in the task often provides ways to simplify and speed up reinforcement learning algorithms and also enables ways to generalize such algorithms over multiple tasks without relearning policies for the entire task. In our approach, we use ideas from dynamical systems to find metastable regions in the state space and associate them with abstract states. We use the spectral clustering algorithm PCCA+ to identify suitable abstractions aligned to the underlying structure. The clustering algorithm also provides connectivity information between abstract states which is helpful in learning decision policies across such states. Skills are defined in terms of the transitions between such abstract states. These skills are independent of the current tasks and we show how they can be efficiently reused across a variety of tasks defined over some common state space. Another major advantage of the approach is that it does not need a prior model of the MDP and it works well even when the MDPs are constructed from sampled trajectories. We empirically demonstrate that our method finds effective skills over a variety of domains. An important contribution of our work is the extension of automated skill acquisition to dynamic domains with an exponential state space such as the Infinite Mario game using function approximation of the state space through CMAC encoding with hashing.

Keywords: Reinforcement Learning; Markov Decision Process; Automated state abstractions; Metastability; Dynamic Systems; Subtasks; Skills; Options

*Both authors contributed equally

1 Motivation and Introduction

Automated discovery of skills or options has been an active area of research and several approaches have proposed for the same. The current methods could be broadly classified into sample trajectory based and partition based methods. One existing approach is to identify bottlenecks from trajectories through state space [4] and cache away policies for reaching such bottle neck states as options. Another approach is to identify strongly connected components of the MDP using clustering methods, spectral or otherwise and to identify *access-states* that connect such components as in [6]. Yet another approach is to identify the structure present in a factored state representation [1] and find options that cause what are normally infrequent changes in state variables. Identifying subtasks using betweeness centrality measures based on a graphical representation of an agent's interaction with its environment is another method that has been tried [9]. While these methods have had varying amounts of success, they have certain deficiencies. Bottleneck based approaches don't have a natural way of identifying the part of the state space where options are applicable without external knowledge about the problem domain. Spectral methods need some form of regularization in order to prevent unequal splits, and this can lead to arbitrary splitting of the state space.

In this paper, we present a framework that detects well-connected or meta stable regions of the state space from a MDP model estimated from trajectories. We use PCCA+, a spectral clustering algorithm from conformal dynamics [11] that not only partitions the MDP but also returns the connectivity information between the regions. We then propose a very effective way of constructing options using the same framework to take us from one metastable region to another. Using these options, we use SMDP Value learning to learn a policy over the subtask to solve the given task. One major advantage of the approach is that we get the policy for free while doing the partitioning by exploiting the membership functions returned by PCCA+. Our approach is able to learn reasonably good skills even with limited sampling which makes it useful in situations where exploration is limited by the environment costs. It also provides a way to refine the abstractions in an online fashion without explicitly reconstructing the entire MDP. More importantly, we extend it to the case where the state space is so large that exact modeling is not possible. In this case, we utilize function approximation through state-aggregation and estimate the transition model on these aggregated states. The approach is shown to work well in practice using such an approximation on the Mario domain.

The main advantages of the PCCA+HRL method are:

1. PCCA+HRL does not require any prior model for the MDP and can acquire skills from sampled trajectories.
2. The clustering algorithm produces characteristic functions that describe the degree of membership for all the states belonging to a particular metastable region and also provide a powerful way to naturally compose options.
3. PCCA+HRL looks for well-connected regions and not bottle-neck states and hence discovers options that are better aligned to the structure of the state space.
4. PCCA+HRL algorithm also returns connectivity information between the metastable regions, which allows us to construct an abstract graph of the state space, where each node is a metastable region thus combining both spatial and temporal abstractions meaningfully.[1]
5. Since the method works with sampled trajectories, PCCA+HRL can be run on abstract state representations.

2 Spatial Abstraction using PCCA+

Spectral clustering was made popular by the works of [8], [5] and [2]. Although the spectra of the Laplacian preserves the structural properties of the graph, clustering data in the eigenspace of the Laplacian does not guarantee this. For example, k-means clustering [7] in the eigenspace of the Laplacian will only work if the clusters lie in disjoint convex sets of the underlying eigenspace. Partitioning the data into clusters by projecting onto the largest k-eigenvectors [5] does not preserve the topological properties of the data in the eigenspace of the Laplacian. For our task of spatial abstraction, we require a clustering approach that exploits the structural properties in the configurational space of objects as well as the spectral sub-space, quite unlike earlier methods. We take inspiration from the work of [11] which proposes a spectral clustering algorithm PCCA+ based on the principles of Perron Cluster Analysis of the transition structure. We extend their analysis to detect spatial abstractions in autonomous controlled dynamical systems.

In this approach, the spectra of the Laplacian L (derived from the adjacency matrix S) is constructed and the best transformation of the spectra is found such that the transformed basis aligns itself with the clusters of data points in the eigenspace. We use a projection method to find the membership of each of the states to a set of special points lying on the transformed basis, which we identify as vertices of a simplex in the $\mathbb{R}^k$ subspace (the Spectral Gap method is used to estimate the number of clusters k). For the first order perturbation, the simplex is just a linear transformation around the origin and to find the simplex vertices, we need to find the k points which form a convex hull such that the deviation of

[1] While it is possible to create such graphs with any set of options using some ad-hoc processing based on domain knowledge, the structure of the graphs so derived depend on the set of options used and may not reflect the underlying spatial structure completely.

all the points from this hull is minimized. Hence, we start by finding the data point which is farthest located from the origin and iteratively identify data points which are located farthest from the hyperplane fit to the current set of vertices. Please refer to [11] for a complete description of the approach.

The PCCA+ algorithm returns a membership function, χ, defining the degree of membership of each state s to an abstract state S_j. The connectivity information between two abstract states (S_i, S_j) is given by $(i,j)^{th}$ entry of $\chi^T L \chi$ while the diagonal entries provide relative connectivity information within a cluster. The connectivity information is utilized to learn decision policies across abstract states. This framework also contains an intrinsic mechanism to return information about the goodness of clustering of states from the presence of sharp peaks (indicates good clustering) in the eigenvalue distribution.

3 Option generation from PCCA+HRL

We use the partitions of the state space into abstract states along with the membership function to define options [10]. We use the structural information obtained to define behavioral policies for the subtasks independent of the task being solved as we believe that to find a hierarchical optimal solution for the entire task, it is not necessary to solve the subtasks optimally and that any behavior interpreted by exploiting the structure present in the domain can be used. This claim is strengthened by our observations on experiments run on several domains. Hence, we do not have to *learn* the option policies, rather these policies are derived from the structures exploited in the state space.

In order to derive subtask options, the membership function χ gives us an elegant method to compose options to exit from a given abstract state. In case of multiple exits or bottlenecks, PCCA+HRL is able to compose multiple options, each taking the agent to the respective exit. Formally, an option is a triple (I,μ,β). Given that the agent is in state s, the **initiation set** I consists of all states belonging to the abstract state S such that $argmax_j \chi_{ij} = argmax_j \chi_{sj} = S \forall i$. Suppose the state s belongs to abstract state S_i, we want to construct a policy to take the agent to the exit of S_i leading to the abstract state S_j. If the agent follows a stochastic gradient ascent on the membership function $m_{S_j}(s) \forall s \in S_i$, then this would take the agent to the exit of the abstract state. Thus, we define the **option policy** $\mu(s,a)$ which takes the agent from S_i to S_j as a stochastic gradient function as follows: $\mu(s,a) = max(\alpha(s)(\sum_{s'} P(s,a,s') m_{S_j}(s') - m_{S_j}(s)), 0) \forall s \in S_i$, where $\alpha(s)$ is a normalization constant to keep the values of $\mu \in [0,1]$. Finally, **termination condition** β is a probability function which assigns the probability of termination of the current option at state s. It can also be viewed as the probability of a state s being a decision epoch given the current option being executed. For an option taking an agent from abstract state S_i to S_j, we define β as follows: $\beta(s) = min(\frac{log(m_{S_i}(s))}{log(m_{S_j}(s))}, 1) \forall s \in S_i$, since it provides a smooth peaked function.

We evaluated our approach on the 2-room domain. The domain consists of 2 unequal sized rooms where the agent has to start from one room and end in the other. We observe that PCCA+HRL acquires the skill to reach the doorway from any state in the first room and to navigate from this doorway to the intended goal state. We identify 3 abstract states where one corresponds to the lone goal state. Figure 1a compares the average return with respect to the number of epochs of decision for different methods. Our approach identifies 2 options as compared to 12 by LCut and 10 by Random Options with primitive actions method, and also consistently attains higher average return than the two.

4 Model Estimation

We now propose an online method for efficiently finding Spatio-Temporal abstractions while the agent is following another strategy. The method is inspired from the UCT framework which is a Monte-Carlo search algorithm based on rollouts. A rollout-based algorithm builds its look-ahead tree by repeatedly sampling episodes from the initial state. The tree is built by adding the information gathered during an episode to it in an incremental manner. We use the UCT algorithm because it is more effective than the vanilla Monte-Carlo planning where the actions are sampled uniformly, while UCT does a selective sampling of actions.

In the UCT approach, in state s, at depth d, the action that maximizes $Q_t(s,a,d) + c_{N_{s,d}(t), N_{s,a,d}(t)}$ is selected, where $Q_t(s,a,d)$ is the estimated value of action a in state s at depth d and time t, $N_{s,d}(t)$ is the number of times state s has been visited up to time t at depth d and $N_{s,a,d}(t)$ is the number of times action a was selected when state s has been visited, up to time t at depth d, $c_{t,s}$ has the form $c_{t,s} = 2C_p \sqrt{\frac{ln(t)}{s}}$, where C_p is an empirical constant. A variant of this search method is used in PCCA+HRL. Since we are composing option policies rather than learning them, we replace $Q(s,a)$ with the stochastic gradient function $\mu(s,a)$, where the particular μ corresponding to the greedy option chosen from the option value function. Hence the search criteria becomes $max(argmax_a(s)(\sum_{s'} P(s,a,s') m_{s_j}(s') - m_{s_j}(s), 0)(d) + c_{N_{s,d)}(t), N_{s,a,d}(t)}$. To include the underlying reward structure, we modify the local adjacency matrix D as $D_{posterior} = D_{prior}(s,s') + \sum_a \phi^a_{ss'} e^{-v|R^{ss'}_a|}$ where $R^{ss'}_a$ is the reward received while v is the regularization constant and $\phi^a_{ss'}$ is the number of times the transition occurred from s to $s's$ with action a which ensures the adjacency function has

very low value at spike points, returns a value for 1 for zero rewards and allows for easy tuning of relative weights by change the parameter v.

Algorithm 1 PCCA+HRL

$Q \Longrightarrow ActionValueFunction$

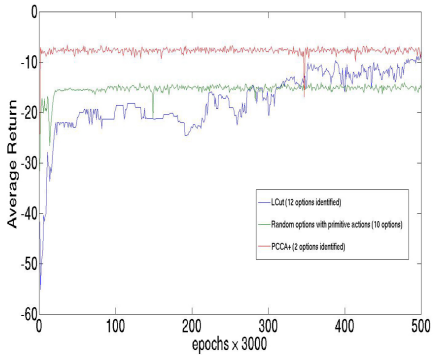
Observe initial state s_o
Initialize Q arbitrarily
Initialize T transition matrix
U={}
for e=1 to maximum number of episodes **do**
    Initialize Membership function χ, Simplex Vertex Y=$PCCA + (\tau)$
    Find all pairs of connected abstract states C_k=(S$_j$, S_k) from the non-zero entries in $\chi^T T \chi$
    $O'_k = O_k \forall k$
    $\forall C_k$ construct $O_k = I_k, \mu_k, \beta_k$
    match $O_k \forall k$(New set of Options) with previous sets of options $O'_k \forall k$
    find k for which $s_o \in C_k(1)$
    $i = k; s_i = s_0$
    while not the end of episode **do**
        $O_i \Leftarrow \begin{cases} argmax_O Q(s_i, O)\ w.p\ 1-\epsilon \\ \text{A uniform random option } O_k = I_k, \mu_k, \beta_k\ w.p\ \epsilon\ s.t\ s_i \in I_k \end{cases}$
        Update $Q(O_i, s_i)$ using any action value function learning method
        Sample action according to $\alpha(\mu_i(s) + c_{N_{s,d}(t), N_{s,a,d}(t)})$ and follow until termination. return termination state s_t
        $\phi^a_{ss'} = \phi^a_{ss'} + \delta^a_{ss'}$ where δ-function =1 if action a takes from state s to s'
        $R^{ss'}_a$ = reward returned while taking action a in state s taking the system to state s'
        $U(s,a,s') = U(s,a,s') + \phi^a_{ss'} e^{-v|R^{ss'}_a|}$
        $D(s,s') = \sum_a U(s,a,s');\ P(s,a,s') = \frac{U(s,a,s')}{\sum_{s'} U(s,a,s')}$
        $T(s,s') = \frac{D(s,s')}{\sum_s D(s,s')}$
        $s_i = s_t$

The proposed approach was evaluated on the taxi domain. The problem can be formulated as an episodic MDP with 3 state variables: location of taxi, passenger location in the taxi and destination location out of 4 designated locations in the world. The problem is episodic wherein in each episode, the taxi starts at a randomly chosen square, there is a passenger at one of the 4 locations (chosen randomly) and the passenger wishes to be transported to the destination. The taxi must go to the passenger, pick up the passenger, go to the destination and put down the passenger. The episode ends when the passenger reaches the destination. There are 6 primitive actions possible, 4 navigation actions that move the taxi one square North, South, East or West, a pickup and a put-down action. There is a rewards of -1 for each action, +20 for delivering the passenger and -10 if the taxi executes pickup or put-down illegally.

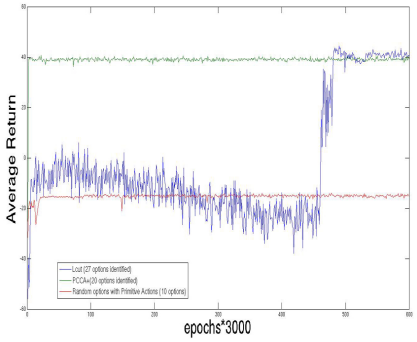
The results for the average return were compared with the LCut method and with Random Options generated using primitive actions in Figure 1b. The PCCA+HRL method identified 20 options while the LCut method identified 27 options. Among the 20 options discovered by PCCA+HRL are the different pick-up and drop-off options corresponding to the different destinations. Our approach consistently performed significantly better than the random options method. In the case of the LCut approach, our approach does significantly better for smaller number of epochs. For larger number of epochs, the LCut method performs marginally better but the difference is not at all significant.

5 PCCA+HRL with function approximation

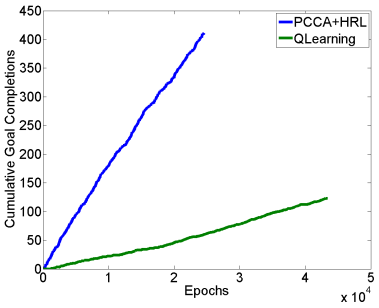
We extend our approach to the case where the state space is so large that exact modeling is not possible. In this case, we perform function approximation through state-aggregation, learn a transitional model in these aggregated states and run partitioning on it. This approach was tested on the Infinite Mario domain (developed for Reinforcement Learning Competition 2009). It is interesting because it requires the the agent to learn at several levels; from representation to path-planning and devising strategies to deal with various components of the environment. The state space has no set specification. The visual scene at any time is divided into a 2-D matrix of 352 tiles, where each tile can take 25 values, leading to 25^{352} possible states at every instant. The agent can choose to stay still or move left or right at two different speeds. It can jump while moving or standing. It earns a huge positive reward when it reaches the finish line and smaller reward by collecting coins, mushrooms and killing monsters. It gets some negative reward by dying before reaching the finish line and a small negative reward for each step taken.



(a) Avg. Reward for 2-room domain



(b) Avg. Return for Taxi domain



(c) Cumulative goal completions in Mario Domain

Since the domain is dynamic, we define coordinates with Mario as a reference point. We define a CMAC encoding with hashing to make the state representation tractable. A CMAC uses multiple overlapping tilings of the state space to produce a feature representation for a linear mapping where all learning takes place. To avoid the curse of dimensionality and reduce memory requirements with little loss of performance through *hashing*, a consistent random collapsing of a large set of tiles into a smaller set. We chose a grid size of 1024 with 2 tilings, giving us a state space of size 4096. Upon running PCCA+HRL to autonomously play Mario, it was able to compose subtasks on structures in the game, like *kill the monster*, *collect the coin*, etc. We compare our results with the QLearning technique in Figure 1c. The performance measure here is the cumulative number of sub-goals achieved over the episodes.

6 Conclusion and Future Work

Viewing random walks on MDPs as dynamical systems allows us to decompose the state space along the lines of temporal scales of change. Thus well-connected regions of the state space were identified as metastable states and we proposed a complete algorithm that identifies metastable states and learns options to navigate between them. We demonstrated the effectiveness of the approach on a variety of domains. We also discussed some crucial advantages of our approach over existing option discovery algorithms. While the approach detects intuitive options, it is possible that under a suitable re-representation of the state space, some of the metastable regions detected can be identical to each other. We are looking to use notions of symmetries in MDPs to identify equivalent metastable regions. We wish to explore function approximation of the state space further in the context of option discovery in the observation space. Another promising line of inquiry is to extend our approach to continuous state spaces, taking cues from Proto Value Functions [3].

References

[1] Bernhard Hengst. Discovering hierarchy in reinforcement learning with hexq. *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 243–250, 2002.

[2] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, May 2004.

[3] Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. *In Proceedings of the International Conference on Machine Learning*, pages 553–560, 2005.

[4] A McGovern. *Discovery of Temporal Abstractions from Intention with an Environment*. PhD thesis, University of Massachusetts - Amherst.

[5] Marina Meila and Jianbo Shi. A Random Walks View of Spectral Segmentation. *Proceedings of the Eighth International Workshop on Artifical Intelligence and Statistics*, 2001.

[6] Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cutdynamic discovery of sub-goals in reinforcement learning. *Machine Learning: ECML 2002*, 2430:295–306, 2002.

[7] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856, 2001.

[8] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.

[9] Ozgur Simsek and Andrew G. Barto. Skill characterization based on betweenness. *In Advances in Neural Information Processing Systems 22*, 2009.

[10] Richard Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[11] M. Weber, W. Rungsarityotin, and A. Schliep. Perron Cluster Analysis and its Connection to Graph Partitioning for Noisy Data. *ZIB Report*, 04-39, 2004.

A Cooperative Game Theoretic Approach to Trajectory Optimization

Yunpeng Pan
Daniel Guggenheim School of Aerospace Engineering
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, GA 30332
ypan37@gatech.edu

Kaivalya Bakshi
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332
kbakshi@gatech.edu

Evangelos A. Theodorou
Daniel Guggenheim School of Aerospace Engineering
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, GA 30332
evangelos.theodorou@ae.gatech.edu

Abstract

While the framework of trajectory optimization based on local approximations of the dynamics and value function has been available for over four decades, it was only recently explored in terms of applicable algorithms for efficient control of real robotic and biological systems. Although local trajectory optimization is more scalable than global optimal control and reinforcement learning methods, it is still challenging to combine computational efficiency and robustness to model errors. Inspired by the work on differential game formulations and their connection to linear stochastic control problems [1], in this work we reformulate a nonlinear stochastic control problem to a Cooperative Stochastic Differential Game (CSDG). Based on second-order local dynamic programming, we introduce a method for solving CSDG called Cooperative Game-Differential Dynamic Programming (CG-DDP). Different from the classical Differential Dynamic Programming (DDP), CG-DDP seeks two cooperative control policies that exhibit more robust performance under stochastic disturbance than a single control policy. Compared to the minimax-DDP which has a non-cooperative game interpretation, the proposed framework features a more efficient optimization scheme. We demonstrate the performance of CG-DDP using two simulated examples.

Keywords: Trajectory optimization, stochastic differential game, optimal control, dynamic programming.

1 Introduction

Model-based trajectory optimization with backward-forward sweeps is a family of powerful frameworks in the field of optimal control. It was originally introduced in 1970 under the name "Differential Dynamic Programming (DDP)" [2]. Since then numerous extensions and variations of DDP have been developed in control, machine learning and robotics communities to improve its performance and applicability [3, 4, 5, 6, 7, 8]. The most attractive characteristics of DDP are its computational efficiency and scalability to high-dimensional dynamical systems compared to global methods. However, since DDP is rooted in local approximation of the dynamics model and value function around a nominal trajectory, model errors could significantly degrade its performance and therefore restrict its applications in many real world scenarios. In control theory the issue of robustness has been addressed in two ways, namely robust control and stochastic control. On one hand, modern robust control methods seek to bound the model uncertainty (H^∞ control). For instance the minimax DDP [3] is an extension of DDP and a conservative (risk sensitive) approach by considering the worst case scenario. While the minimax-DDP is theoretically appealing due to the explicitly bounded disturbance tolerance, numerical convergence to the desired saddle point is challenging. On the other hand, stochastic control takes into account noise as probability distributions for control law design. In this class of methods a typical assumption is zero-mean Gaussian noises.

In this work, we present an alternative approach to trajectory optimization based on Cooperative Stochastic Differential Games, called Cooperative Game-Differential Dynamic Programming (CG-DDP). Different from existing trajectory optimization frameworks, CG-DDP seeks two cooperative control policies. We will demonstrate its performance under disturbance using numerical examples.

2 Cooperative Stochastic Differential Game Problem Formulation

Consider the following jump diffusion process given by the stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x} = \mathbf{f}(t,\mathbf{x}(t))\mathrm{d}t + \mathbf{B}(t,\mathbf{x}(t))\boldsymbol{\tau}(t)\mathrm{d}t + \mathbf{C}(t)\mathrm{d}\boldsymbol{\omega} + \mathbf{H}(t)\mathrm{d}\mathbf{p}, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is the state, $\boldsymbol{\tau} \in \mathbb{R}^{n_u}$ is the control, $\boldsymbol{\omega} \in \mathbb{R}^{n_w}$ and $\mathbf{p} \in \mathbb{R}^{n_p}$ are standard Wiener and Poisson processes. The dimensionality of the terms $\mathbf{f}, \mathbf{B}, \mathbf{C}, \mathbf{H}$ in (1) is defined as $\mathbf{f} \in \mathbb{R}^{n_x}$, $\mathbf{B} \in \mathbb{R}^{n_x \times n_u}$, $\mathbf{C} \in \mathbb{R}^{n_x \times n_w}$ and $\mathbf{H} \in \mathbb{R}^{n_x \times n_p}$. Moreover let $\mathbf{h}^j$ be the j^{th} the column vector in $\mathbf{H}$. For the Poisson process $\mathbb{E}[\mathrm{d}\mathbf{p}^j] = \lambda^j \mathrm{d}t$ and $\lambda^j \in \mathbb{R}\ \forall 1 \leq j \leq n_p$. We denote $\boldsymbol{\lambda} = [\lambda]_{n_p \times 1}$ the rate with which jumps occur. Consider a finite-horizon stochastic optimal control problem for the cost functional

$$J(t,\mathbf{x};\pi) = \mathbb{E}\Big[\underbrace{q(\mathbf{x}(T))}_{\textbf{Terminal cost}} + \int_t^T \underbrace{\mathcal{L}\big(t,\mathbf{x}(t),\pi(t,\mathbf{x}(t))\big)}_{\textbf{Running cost}} \mathrm{d}t \Big], \tag{2}$$

where the goal is to find a control policy $\boldsymbol{\tau} = \pi(t,\mathbf{x}(t))$ that minimizes the total cost accumulated over the time interval $[t,T]$. Here we define two control variables $\mathbf{u} \in \mathbb{R}^{n_u}$ and $\mathbf{v} \in \mathbb{R}^{n_u}$ such that the original control input is split into two parts: $\boldsymbol{\tau} = \mathbf{u} + \mathbf{v}$. Now we will show that the original control problem can be reformulated as a two-player Cooperative Stochastic Differential Game with a given time interval $[t,T]$ and players $\mathbf{u}$,$\mathbf{v}$. Let functions $\pi^\mathbf{u}$, $\pi^\mathbf{v}$ be strategies (policies) for $\mathbf{u}$ and $\mathbf{v}$ from t to T. $\pi^\mathbf{u} = \emptyset$ implies $\mathbf{u}(t),...,\mathbf{u}(T) = 0$ and $\pi^\mathbf{v} = \emptyset$ is defined similarly. In addition $\pi^u \cup \pi^v$ denotes a cooperative game theoretic control strategy. Next we define the payoff function for each individual player. For player $\mathbf{u}$ the payoff $\mathcal{V}^\mathbf{u}(\pi^\mathbf{u} \cup \pi^\mathbf{v}) = J^\mathbf{u}(t,\mathbf{x};\emptyset) - J^\mathbf{u}(t,\mathbf{x};\pi^\mathbf{u},\pi^\mathbf{v})$, $\forall \pi^\mathbf{u}, \pi^\mathbf{v}$, where $J^\mathbf{u}(t,\mathbf{x};\pi^\mathbf{u},\pi^\mathbf{v}) = \mathbb{E}\Big[q^\mathbf{u}(\mathbf{x}(T)) + \int_t^T \mathcal{L}^\mathbf{u}(t,\mathbf{x}(t),\pi^\mathbf{u},\pi^\mathbf{v})\mathrm{d}t\Big]$, and $J^\mathbf{u}(t,\mathbf{x};\emptyset)$ corresponds to the cost with both controls taken to be zero. Payoff for $\mathbf{v}$, i.e., $\mathcal{V}^\mathbf{v}(\pi^\mathbf{u} \cup \pi^\mathbf{v})$ can be defined similarly. The coalition or cooperative payoff function is defined as $\mathcal{V}(\pi^\mathbf{u} \cup \pi^\mathbf{v}) = J(t,\mathbf{x};\emptyset) - J(t,\mathbf{x};\pi^\mathbf{u} \cup \pi^\mathbf{v})$, where $J(t,\mathbf{x};\pi^\mathbf{u} \cup \pi^\mathbf{v}) = \mathbb{E}\Big[q(\mathbf{x}(T)) + \int_t^T \mathcal{L}(t,\mathbf{x}(t),\pi^\mathbf{u},\pi^\mathbf{v})\mathrm{d}t\Big]$, and $J(t,\mathbf{x};\emptyset)$ is the uncontrolled cost. Therefore the coalition $\pi^\mathbf{u} \cup \pi^\mathbf{v}$ has a real valued payoff assigned by the cooperative payoff function for which $\mathcal{V}(\emptyset) = 0$. Both $\mathbf{u}$ and $\mathbf{v}$ try to maximize the assigned individual payoff function at t. The above stated problem defines a coalition game with players $\mathbf{u}$ and $\mathbf{v}$. We follow [9] and denote this two player *C*ooperative *G*ame by *CG*$(\mathbf{x}, T-t)$ played at current state $\mathbf{x}$ for the time interval $[t,T]$, where both players $\mathbf{u}$ and $\mathbf{v}$ agree to cooperate based on an optimality principle. The optimality principle for a cooperative scheme includes: i) an agreement on a set of cooperative strategies/controls, and ii) a mechanism to distribute total payoff among players, iii) group rationality requires the players to seek a set of optimal cooperative strategies $\pi^\mathbf{u}$ and $\pi^\mathbf{v}$ that justify the participation of each in the *CG*$(\mathbf{x}, T-t)$. This is mathematically expressed as $\mathcal{V}(\pi^{\mathbf{u}*} \cup \pi^{\mathbf{v}*}) \geq \mathcal{V}(\emptyset \cup \pi^\mathbf{v}), \mathcal{V}(\pi^\mathbf{u} \cup \emptyset)$, $\forall \pi^\mathbf{u}, \pi^\mathbf{v}$. Considering the fact that $J(t,\mathbf{x};\emptyset)$ does not depend on the selected strategies, the solution to *CG*$(\mathbf{x}, T-t)$ is therefore to solve $\underset{\pi^\mathbf{u},\pi^\mathbf{v}}{\operatorname{argmin}}\, \mathbb{E}\Big[\sum_{j=\{\mathbf{u},\mathbf{v}\}} q^j(\mathbf{x}(T)) + \int_t^T \sum_{j=\{\mathbf{u},\mathbf{v}\}} \mathcal{L}^j(t,\mathbf{x}(t),\pi^\mathbf{u},\pi^\mathbf{v})\mathrm{d}t\Big]$. We choose to distribute the individual payoff functions s.t. $q^\mathbf{u}(\mathbf{x}) + q^\mathbf{v}(\mathbf{x}) = q(\mathbf{x}), \quad \mathcal{L}^\mathbf{u}(\mathbf{x}) + \mathcal{L}^\mathbf{v}(\mathbf{x}) = \mathcal{L}(\mathbf{x})$, so that we seek strategies $\pi^\mathbf{u}, \pi^\mathbf{v}$ to solve

$$\underset{\pi^\mathbf{u},\pi^\mathbf{v}}{\operatorname{argmin}}\, \mathbb{E}\Big[q(\mathbf{x}(T)) + \int_t^T \mathcal{L}(t,\mathbf{x}(t),\pi^\mathbf{u},\pi^\mathbf{v})\mathrm{d}t\Big], \tag{3}$$

subject to (1). This problem is called the *C*ooperative *S*tochastic *D*ifferential *G*ame in Game Theory literature [9], and is denoted by *CSDG*$(\mathbf{x}, T-t)$ the solution to which is stated in the following theorem.

Theorem 2.1 *A set of controls $[\mathbf{u}^*(t), \mathbf{v}^*(t)]$ provides an optimal solution to the problem CSDG$(\mathbf{x}, T-t)$ if there exists a continuously differentiable function $V(t,\mathbf{x}) : [t,T] \times \mathbb{R}^{n_x} \to \mathbb{R}$ which satisfies the PDE*

$$-\frac{\partial V}{\partial t}(t,\mathbf{x}) - \frac{1}{2}\mathrm{tr}(\mathbf{C}\mathbf{C}^{\mathrm{T}} V_{\mathbf{xx}}(t,\mathbf{x})) - \sum_{j=1}^{n_p}(V(t,\mathbf{x}+\mathbf{h}^j(t)) - V(t,\mathbf{x}))\lambda^j(t) = \min_{\mathbf{u}(t),\mathbf{v}(t)}\Big(\mathcal{L}(t,\mathbf{x},\mathbf{u},\mathbf{v}) + V_{\mathbf{x}}^T(\mathbf{f}(\mathbf{x}) + \mathbf{B}(\mathbf{x})(\mathbf{u}+\mathbf{v})\Big),$$

with the boundary condition $V(T,\mathbf{x}) = q(T,\mathbf{x})$.

Theorem 2.1 is an extension of [9] for the case of minimum game for jump diffusion processes. This theorem provides necessary conditions for the existence of cooperative control strategies that are group rational. Given the existence of such a value function one has to solve this PDE in order to obtain the optimal control strategy at a given $(t,\mathbf{x})$. However, finding the solution for the PDE in Theorem 2.1 is computationally intractable especially for systems with moderate to high dimensions. This is so-called the *curse of dimensionality*. One way to bypass this is to rely on approximation methods that solve this problem locally, in particular by approximating the value function along nominal trajectories. DDP is known to be one of the most efficient approaches to value function approximation that features second order convergence and scales to high dimensional systems. In this work we propose a scheme to solve the CSDG locally by transforming the continuous-time problem to discrete-time and working with second order approximations of value/cost functions, and first order approximation of the dynamics. The resulting framework is called Cooperative Game-Differential Dynamic Programming (CG-DDP). For the rest of the analysis we use the discretized representation of the system in (1), i.e.,

$$\mathbf{x}_{t+\mathrm{d}t} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t)\mathrm{d}t + \mathbf{B}(\mathbf{x}_t)(\mathbf{u}_t+\mathbf{v}_t)\mathrm{d}t + \mathbf{C}(t)\mathrm{d}\boldsymbol{\omega} + \mathbf{H}(t)\mathrm{d}\mathbf{p}. \tag{4}$$

To simplify notation we use subscripts of variables to denote discrete time step, e.g., $\mathbf{x}_t = \mathbf{x}(t)$.

3 The Cooperative Game-Differential Dynamic Programming Framework

3.1 Backward-sweep: optimal cost-to-go

The proposed trajectory optimization method is rooted in the Dynamic Programming principle. First we consider a variation of the Bellman equation with both $\mathbf{u}_t$ and $\mathbf{v}_t$

$$V(t,\mathbf{x}_t) = \min_{\mathbf{u}_t,\mathbf{v}_t} \underbrace{\mathbb{E}\Big[\mathcal{L}(t,\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t) + V(t+\mathrm{d}t,\mathbf{x}_{t+\mathrm{d}t})\Big]}_{Q(\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t)}. \tag{5}$$

The goal is to find two cooperative feedback policies such that the pair $(\mathbf{u}_t,\mathbf{v}_t)$ minimizes the total cost. First we build a local model of the dynamics model along a nominal trajectory $\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t, \bar{\mathbf{v}}_t$.

$$\delta\mathbf{x}_{t+\mathrm{d}t} = \underbrace{\mathbf{A}_t\delta\mathbf{x}_t + \mathbf{B}_t\big(\delta\mathbf{u}_t + \delta\mathbf{v}_t\big) + \mathbf{C}_t\mathrm{d}\boldsymbol{\omega}}_{\delta\mathbf{x}^{\text{cont}}_{t+\mathrm{d}t}} + \underbrace{\mathbf{H}_t\mathrm{d}\mathbf{p}}_{\delta\mathbf{x}^{\text{jump}}_{t+\mathrm{d}t}}, \tag{6}$$

where $\delta\mathbf{x}_t = \mathbf{x}_t - \bar{\mathbf{x}}_t, \delta\mathbf{u}_t = \mathbf{u}_t - \bar{\mathbf{u}}_t$ and $\delta\mathbf{v}_t = \mathbf{v}_t - \bar{\mathbf{v}}_t$ are defined as the deviations from the nominal trajectory, $\mathbf{A}_t = (I_{n_x\times n_x} + \mathbf{f}_{\mathbf{x}}(\bar{\mathbf{x}}_t))\mathrm{d}t$, $\mathbf{B}_t = \mathbf{B}(t,\bar{\mathbf{x}}_t)\mathrm{d}t$, $\mathbf{C}_t = \mathbf{C}(t)\sqrt{\mathrm{d}t}$ and $\mathbf{H}_t = \mathbf{H}(t)\mathrm{d}t$. To evaluate the expectation under minimization in (5) we use the Ito stochastic chain rule for jump diffusion process and the nonlinear jump term of the value function has been approximated till its second order Taylor series expansion. Next we build a local quadratic model of the value function by expanding the Q-function up to the second order

$$Q(\bar{\mathbf{x}}_t+\delta\mathbf{x}_t, \bar{\mathbf{u}}_t+\delta\mathbf{u}_t, \bar{\mathbf{v}}_t+\delta\mathbf{v}_t) \approx Q_0 + Q_{\mathbf{x}}\delta\mathbf{x}_t + Q_{\mathbf{u}}\delta\mathbf{u}_t + Q_{\mathbf{v}}\delta\mathbf{v}_t + \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\\ \delta\mathbf{v}_t\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}Q_{\mathbf{xx}} & Q_{\mathbf{xu}} & Q_{\mathbf{xv}}\\ Q_{\mathbf{ux}} & Q_{\mathbf{uu}} & Q_{\mathbf{uv}}\\ Q_{\mathbf{vx}} & Q_{\mathbf{vu}} & Q_{\mathbf{vv}}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\\ \delta\mathbf{v}_t\end{bmatrix}. \tag{7}$$

where all the Q-related terms are given as

$$\begin{aligned}
&Q_0 = V_{t+\mathrm{d}t} + \frac{1}{2}\mathrm{tr}(V_{\mathbf{xx}}\mathbf{C}_t\mathbf{C}_t^{\mathrm{T}}) + \sum_{j=1}^{n_p}\Big(V_{\mathbf{x}}^{\mathrm{T}}\mathbf{h}_t^j\lambda_t^j + (\mathbf{h}_t^j)^{\mathrm{T}}V_{\mathbf{xx}}\mathbf{h}_t^j\lambda_t^j\Big)\mathrm{d}t,\\
&Q_{\mathbf{x}} = V_{\mathbf{x}}^{\mathrm{T}}\mathbf{A}_t + \mathcal{L}_{\mathbf{x}},\ Q_{\mathbf{u}} = V_{\mathbf{x}}^{\mathrm{T}}\mathbf{B}_t + \mathcal{L}_{\mathbf{u}},\quad Q_{\mathbf{v}} = V_{\mathbf{x}}^{\mathrm{T}}\mathbf{B}_t + \mathcal{L}_{\mathbf{v}},\quad Q_{\mathbf{xx}} = \mathbf{A}_t^{\mathrm{T}}V_{\mathbf{xx}}\mathbf{A}_t + \mathcal{L}_{\mathbf{xx}},\quad Q_{\mathbf{xu}} = \mathbf{A}^{\mathrm{T}}V_{\mathbf{xx}}\mathbf{B}_t + \mathcal{L}_{\mathbf{xu}},\\
&Q_{\mathbf{xv}} = \mathbf{A}_t^{\mathrm{T}}V_{\mathbf{xx}}\mathbf{B}_t + \mathcal{L}_{\mathbf{xv}},\quad Q_{\mathbf{uu}} = \mathbf{B}_t^{\mathrm{T}}V_{\mathbf{xx}}\mathbf{B}_t + \mathcal{L}_{\mathbf{uu}},\quad Q_{\mathbf{vv}} = \mathbf{B}_t^{\mathrm{T}}V_{\mathbf{xx}}\mathbf{B}_t + \mathcal{L}_{\mathbf{vv}},\quad Q_{\mathbf{uv}} = \mathbf{B}_t^{\mathrm{T}}V_{\mathbf{xx}}\mathbf{B}_t + \mathcal{L}_{\mathbf{uv}},
\end{aligned} \tag{8}$$

where $V_t = V(t,\bar{\mathbf{x}}_t)$. In order to find the optimal policies for $\delta\mathbf{u}_t$ and $\delta\mathbf{v}_t$ such that the second-order expansion of the Q-function is minimized, we take the gradients of (7) with respect to $\delta\mathbf{u}_t$ and $\delta\mathbf{v}_t$

$$\begin{aligned}
Q_{\delta\mathbf{u}}\big(\mathbf{x}_t+\delta\mathbf{x}_t, \mathbf{u}_t+\delta\mathbf{u}_t, \mathbf{v}_t+\delta\mathbf{v}_t\big) = 0 \quad &\Rightarrow \quad \delta\mathbf{u}_t^* = -Q_{\mathbf{uu}}^{-1}\big(Q_{\mathbf{ux}}\delta\mathbf{x}_t + Q_{\mathbf{uv}}\delta\mathbf{v}_t + Q_{\mathbf{u}}\big),\\
Q_{\delta\mathbf{v}}\big(\mathbf{x}_t+\delta\mathbf{x}_t, \mathbf{u}_t+\delta\mathbf{u}_t, \mathbf{v}_t+\delta\mathbf{v}_t\big) = 0 \quad &\Rightarrow \quad \delta\mathbf{v}_t^* = -Q_{\mathbf{vv}}^{-1}\big(Q_{\mathbf{vx}}\delta\mathbf{x}_t + Q_{\mathbf{vu}}\delta\mathbf{u}_t + Q_{\mathbf{v}}\big).
\end{aligned} \tag{9}$$

Solving the system of equations (9) results in explicit expressions of optimal control updates $\delta\mathbf{u}_t = \mathbf{I_u} + \mathbf{L_u}\delta\mathbf{x}_t$ and $\delta\mathbf{v}_t = \mathbf{I_v} + \mathbf{L_v}\delta\mathbf{x}_t$ where $\mathbf{I_u}, \mathbf{I_v}$ and $\mathbf{L_u}, \mathbf{L_v}$ are feedforward and feedback components, respectively. By plugging them into the value function, we can split the value function into zero, first and second order terms in $\delta\mathbf{x}_t$, i.e., $V(\bar{\mathbf{x}}_t + \delta\mathbf{x}_t) = V_t + V_{\mathbf{x}}^{\mathrm{T}}\delta\mathbf{x}_t + \frac{1}{2}\delta\mathbf{x}_t^{\mathrm{T}} V_{\mathbf{xx}}\delta\mathbf{x}_t$, where

$$\begin{aligned} V_t =& Q_0 + \mathbf{l_u}^{\mathrm{T}} Q_{\mathbf{u}} + \mathbf{l_v}^{\mathrm{T}} Q_{\mathbf{v}} + \frac{1}{2}\Big(\mathbf{l_u}^{\mathrm{T}} Q_{\mathbf{uu}}\mathbf{l_u} + \mathbf{l_v}^{\mathrm{T}} Q_{\mathbf{vv}}\mathbf{l_v} + \mathbf{l_u}^{\mathrm{T}} Q_{\mathbf{uu}}\mathbf{l_v} + \mathbf{l_v}^{\mathrm{T}} Q_{\mathbf{vu}}\mathbf{l_u}\Big), \\ V_{\mathbf{x}} =& Q_{\mathbf{x}} + \mathbf{L_u}^{\mathrm{T}} Q_{\mathbf{u}} + \mathbf{L_v}^{\mathrm{T}} Q_{\mathbf{v}} + Q_{\mathbf{xu}}\mathbf{l_u} + Q_{\mathbf{xv}}\mathbf{l_v} + \mathbf{L_u}^{\mathrm{T}} Q_{\mathbf{uu}}\mathbf{l_u} + \mathbf{L_v}^{\mathrm{T}} Q_{\mathbf{vv}}\mathbf{l_v} + \mathbf{L_u}^{\mathrm{T}} Q_{\mathbf{uv}}\mathbf{l_v} + \mathbf{L_v}^{\mathrm{T}} Q_{\mathbf{vu}}\mathbf{l_u}, \\ V_{\mathbf{xx}} =& Q_{\mathbf{xx}} + \mathbf{L_{ux}}^{\mathrm{T}} Q_{\mathbf{ux}} + \mathbf{L_v}^{\mathrm{T}} Q_{\mathbf{v}} + Q_{\mathbf{xu}}\mathbf{l_u} + Q_{\mathbf{xv}}\mathbf{l_v} + \mathbf{L_u}^{\mathrm{T}} Q_{\mathbf{uu}}\mathbf{L_u} + \mathbf{L_v}^{\mathrm{T}} Q_{\mathbf{vv}}\mathbf{L_v} + \mathbf{L_u}^{\mathrm{T}} Q_{\mathbf{uv}}\mathbf{L_v} + \mathbf{L_v}^{\mathrm{T}} Q_{\mathbf{vu}}\mathbf{L_u}. \end{aligned} \tag{10}$$

Since the optimal cost-to-go is computed backward in time, this computational scheme is called the backward-sweep in trajectory optimization. Next we introduce the forward-sweep part of CG-DDP.

3.2 Update control laws and forward sweep

Now we compute the optimal controllers for the next iteration as

$$\mathbf{u}_t^* = \mathbf{u}_t + \delta\mathbf{u}_t^* = \mathbf{u}_t + \mathbf{I_u} + \mathbf{L_u}\delta\mathbf{x}_t, \quad \mathbf{v}_t^* = \mathbf{v}_t + \delta\mathbf{v}_t^* = \mathbf{v}_t + \mathbf{I_v} + \mathbf{L_v}\delta\mathbf{x}_t. \tag{11}$$

Obviously these are linear, time-varying policies but without any a-priori policy parameterization. The results in eq.(11) are locally optimal controls for the original nonlinear system in the vicinity of the nominal trajectory $\bar{\mathbf{x}}$. To obtain an optimal trajectory we iteratively update the nominal trajectory by applying the optimized controls. The resulting trajectory becomes the new nominal for the next iteration. The proposed trajectory optimization approach is a second-order method that relies on the Hessian matrices and are expressed as

$$\mathbf{H_u} = Q_{\mathbf{uu}} - Q_{\mathbf{uv}}Q_{\mathbf{vv}}^{-1}Q_{\mathbf{vu}}, \quad \mathbf{H_v} = Q_{\mathbf{vv}} - Q_{\mathbf{vu}}Q_{\mathbf{uu}}^{-1}Q_{\mathbf{uv}}. \tag{12}$$

The cost-to-go decreases in the direction of $\delta\mathbf{u}_t$ and $\delta\mathbf{v}_t$ for positive definite $\mathbf{H_u}$ and $\mathbf{H_v}$. In the proposed framework we implement line search by adding a parameter $\varepsilon > 0$ such that $\delta\mathbf{u}_t^* = \varepsilon\mathbf{I_u} + \mathbf{L_u}\delta\mathbf{x}_t$ and $\delta\mathbf{v}_t^* = \varepsilon\mathbf{I_v} + \mathbf{L_v}\delta\mathbf{x}_t$. Initially $\varepsilon = 1$, when the trajectory generated by the new policy has a higher cost than the current one, the policy would be rejected and decrease ε. Whenever the policy is accepted we reset $\varepsilon = 1$. This trick has also been used in [4] to encourage convergence. The optimization is performed iteratively in internal simulation. The control policy implemented in the physical system under stochastic disturbances is

$$\boldsymbol{\tau}_t^* = \pi(\mathbf{x}_t) = \underbrace{\bar{\mathbf{u}}_t^* + \mathbf{L_u}\big(\mathbf{x}_t - \bar{\mathbf{x}}_t^*\big)}_{\mathbf{u}_t^*} + \underbrace{\bar{\mathbf{v}}_t^* + \mathbf{L_v}\big(\mathbf{x}_t - \bar{\mathbf{x}}_t^*\big)}_{\mathbf{v}_t^*}. \tag{13}$$

Where $\bar{\mathbf{x}}_t^*, \bar{\mathbf{u}}_t^*, \bar{\mathbf{v}}_t^*$ are the optimized trajectory and controllers obtained from simulation (the final nominal trajectory). $\bar{\mathbf{x}}_t$ is the actual state deviation from the optimal trajectory. We do not apply the open-loop policy $\mathbf{I_u}$ and $\mathbf{I_v}$ whose magnitudes usually vanish (or become very small) during the final stage of optimization.

3.3 Relations to existing methods

The proposed framework is related to various existing methods. In particular, we compare the proposed CG-DDP with DDP [2] and minimax DDP [3] in this section. The proposed CG-DDP is derived similarly as the original DDP [2]. However, CG-DDP is based on a different problem formulation (5) which leads to two cooperative control policies. Note that when the eigenvalues of $Q_{\mathbf{vv}}$ are sufficiently large, the CG-DDP policy is equivalent to the standard DDP policy. For a simple example, in the scalar case we have

$$\begin{aligned} &Q_{\mathbf{vv}} \to \infty \Longrightarrow \mathbf{H_v} \to \infty \Longrightarrow \mathbf{I_v}, \mathbf{L_v} \to 0 \Longrightarrow \delta\mathbf{v}_t^* \to 0, \\ &Q_{\mathbf{vv}} \to \infty \Longrightarrow \mathbf{H_u} \to Q_{\mathbf{uu}} \Longrightarrow \mathbf{I_v} \to Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}}, \mathbf{L_v} \to Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}} \Longrightarrow \delta\mathbf{u}_t^* \to \underbrace{Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}} + Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}}\delta\mathbf{x}_t}_{\textbf{DDP policy}}. \end{aligned}$$

Intuitively when $\mathbf{v}$ is sufficiently expensive, we retrieve the DDP solution. Therefore CG-DDP can be viewed as a generalization of DDP. Furthermore, according to the group rationality of *CSDG*, CG-DDP yields no worse solution than DDP.

One might notice that our approach is related to the minimax DDP [3]. The minimax DDP is derived from a non-cooperative game formula

$$\min_{\mathbf{u}_t}\max_{\mathbf{v}_t}\Big[q(\mathbf{x}_T) + \int_t^T \mathcal{L}\big(t, \mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t\big)\mathrm{d}t\Big]. \tag{14}$$

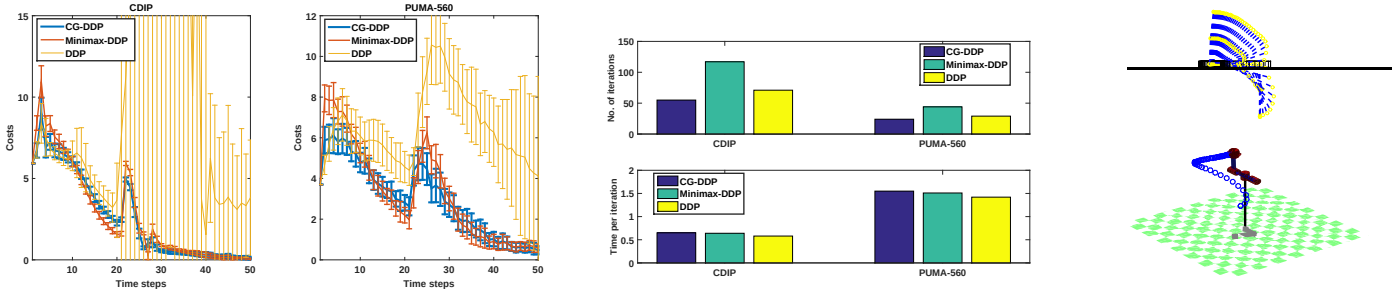
While the scheme is different from our case, the resulting policy formulations share some similarities. In particular eq.(9) appears in the minimax-DDP as well. The minimax DDP is based on H^∞ control theory such that the optimal control gain would minimize the effects of the worst disturbance to the system. There are several key differences between CG-DDP and minimax DDP. In particular, in minimax-DDP the non-cooperative policy $\mathbf{v}$ is not applied to the physical systems since it is treated as disturbances. In CG-DDP the policies for both player $\mathbf{u}, \mathbf{v}$ are applied. Furthermore, in CG-DDP the backward Riccati equations are different from the minimax-DDP case. In particular the coupling terms between $\mathbf{u}$ and $\mathbf{v}$ (e.g.,$Q_{\mathbf{uv}}$) and noise-related terms appear in CG-DDP (10). Compared to minimax-DDP, the major benefits of CG-DDP can be summarized as follows: i) in minimax-DDP the existence of solution depends

on tuning of the cost function. For instance for a cost defined as $\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t) = (\mathbf{x}_t - \mathbf{x}_t^d)^{\mathrm{T}}\mathbf{Q}(\mathbf{x}_t - \mathbf{x}_t^d) + \mathbf{u}_t^{\mathrm{T}}\mathbf{R}_{\mathbf{u}}\mathbf{u}_t - \mathbf{v}_t^{\mathrm{T}}\mathbf{R}_{\mathbf{v}}\mathbf{v}_t$, the existence of the min-max solution depends on the choice of $\mathbf{R}_{\mathbf{u}}$ and $\mathbf{R}_{\mathbf{v}}$. In CG-DDP, the existence of solution does not depend on the tuning of the cost functions. ii) In minimax-DDP the optimal solution is a saddle point while in CG-DDP the optimal solution is an extremum. Numerically it is more challenging to find the saddle point since monotonicity during convergence cannot be assured in the min-max case where $\mathbf{H}_{\mathbf{u}}$ and $\mathbf{H}_{\mathbf{v}}$ are positive definite and negative definite, respectively.

4 Experiments and Analysis

In this section we provide simulation results and analysis based on two dynamical systems: the cart and double inverted pendulum (CDIP) system and the PUMA-560 robot manipulator. For the CDIP (3DOF) system, the task is to swing-up the two-link pendulum from the initial position (both point down). For PUMA-560 (6DOF), the task is to steer the end-effector to the desired position and orientation. See Fig.1d for an example of both tasks. We performed two experiments in order to test the efficiency and robustness of CG-DDP, respectively. For comparison we use the minimax-DDP [3] and standard DDP [2]. For both tasks, we apply non-Gaussian disturbances (jump-diffusion processes) to the systems.

In this experiment we sample 100 stochastic trajectories by applying the optimized control policies for both tasks. Results of trajectory costs are shown in Fig.(1a,1b). Both CG-DDP and minimax-DDP show superior performance in terms of robustness against disturbance as they successfully steer all trajectories to the targets. In the standard DDP case, a few trajectories diverge from the target due to the stochastic disturbances. The performance differences between CG-DDP and DDP can be theoretically justified from the group rationality of the cooperative controllers (solution to CSDG) which guarantees that the two-controller coalition works better (or at least equal) than one. Fig.1c shows the comparison in terms of number of iterations required for convergence and average computational time per iteration. CG-DDP shows faster convergence in both tasks. The minimax-DDP shows slower convergence than the other two methods due to the lack of monotonicity in convergence.



(a) CDIP costs. (b) PUMA-560 costs. (c) Comparison of efficiency. (d) Tasks.

Figure 1: Simulation results

5 Conclusions

Our work introduces a novel trajectory optimization framework based on Cooperative Stochastic Differential Games (CSDG) for nonlinear systems. The resulting framework is called Cooperative Game-Differential Dynamic Programming (CG-DDP). CG-DDP can be considered as a generalization of DDP with two controllers and yields no worse solution than DDP according to the property of group rationality. CG-DDP can be easily integrated with learned dynamics to develop scalable and robust reinforcement learning algorithms. This work can be a precursor to several interesting theoretical problems and applicable algorithms for decision-making in autonomous systems. Future work will focus on extending CG-DDP to the case of systems with unknown dynamics.

References

[1] D.H Jacobson. Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games. *IEEE Transactions on Automatic Control*, 18(2):124–131, 1973.

[2] D.H. Jacobson and D.Q Mayne. *Differential dynamic programming*. Elsevier Sci. Publ., 1970.

[3] J. Morimoto and CG Atkeson. Minimax differential dynamic programming: An application to robust biped walking. In *NIPS*, pages 1539–1546, 2002.

[4] E. Todorov and W. Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *ACC*, pages 300–306, 2005.

[5] P. Abbeel, A. Coates, M. Quigley, and A. Y Ng. An application of reinforcement learning to aerobatic helicopter flight. *NIPS*, 19:1, 2007.

[6] D. Mitrovic, S. Klanke, and S. Vijayakumar. Adaptive optimal feedback control with learned internal dynamics models. In *From Motor Learning to Interaction Learning in Robots*.

[7] J. Van Den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using differential dynamic programming in belief space. In *Intl Symposium on Robotics Research*, 2011.

[8] S. Levine and V. Koltun. Variational policy search via trajectory optimization. In *NIPS*, pages 207–215. 2013.

[9] D.W. Yeung and L.A. Petrosjan. *Cooperative stochastic differential games*. Springer Science & Business Media, 2006.

Parameter Selection for the Deep Q-Learning Algorithm

Nathan Sprague
Department of Computer Science
James Madison University
Harrisonburg, VA 22801
spragunr@jmu.edu

Abstract

Over the last several years deep learning algorithms have met with dramatic successes across a wide range of application areas. The recently introduced deep Q-learning algorithm represents the first convincing combination of deep learning with reinforcement learning. The algorithm is able to learn policies for Atari 2600 games that approach or exceed human performance. The work presented here introduces an open-source implementation of the deep Q-learning algorithm and explores the impact of a number of key hyper-parameters on the algorithm's success. The results suggest that, at least for some games, the algorithm is very sensitive to hyper-parameter selection. Within a narrow-window of values the algorithm reliably learns high-quality policies. Outside of that narrow window, learning is unsuccessful. This brittleness in the face of hyper-parameter selection may make it difficult to extend the use deep Q-learning beyond the Atari 2600 domain.

Keywords: Reinforcement learning, deep Q-Learning, feature learning

1 Introduction

The last several years have seen dramatic progress in the application of deep neural network architectures to problems in both supervised and unsupervised learning. The recently introduced deep Q-learning algorithm [6, 7] represents the first convincing application of deep feature learning to a non-trivial reinforcement learning task. Deep Q-Networks are able to approach or exceed human performance on a range of Atari 2600 games. Policies are learned completely from scratch based on pixel-level input.

The success of deep Q-learning in the Atari domain is a potentially important result. Scaling up reinforcement learning algorithms to handle continuous and high-dimensional tasks has been a longstanding challenge. Powerful feature learning algorithms are likely to be essential to making progress in this area. The success of deep Q-learning is also somewhat surprising. The algorithm has no theoretical convergence guarantees. On the contrary, it is well recognized that Q-learning has a tendency to be unstable when coupled with non-linear function approximation [8, 4]. These factors make it particularly important to reproduce the deep Q-learning results.

The goal of the work described here is twofold: first to reproduce the results of the original deep Q-learning workshop paper [6] by developing an open-source implementation that can be used as a starting point for future research[1], and second, to systematically explore the impact of several key hyper-parameters on the success of the algorithm.

2 Methods

The implementation described here uses the Arcade Learning Environment [2] as an interface to an Atari 2600 emulator. The deep Q-learning implementation is built on top of Theano [3, 1] and uses the neural network code developed for Sander Dieleman's galaxy zoo Kaggle competition entry [5].

The implementation follows the published description as closely as possible. The network architecture, image preprocessing, exploration schedule etc., all match the published specifications. As in the original paper, weight updates are handled using RMSProp with a batch size of 32.

The RMSProp algorithm [9] involves scaling weight updates on a per-weight basis according to a running average of the square of the gradient. The following two equations describe the update rules for tracking the average gradient values and updating weights.

$$r_{t,w} = \rho\, r_{t-1,w} + (1-\rho)\left(\frac{\partial L}{\partial w}\right)^2 \tag{1}$$

$$w_{t+1} = w_t + \alpha \frac{1}{\sqrt{r_{t,w}}} \frac{\partial L}{\partial w} \tag{2}$$

The two hyper-parameters that appear in the equations above are the decay rate ρ and the step size parameter α. The results presented below will explore the impact of these two parameters along with the discount rate γ

An additional factor that is not described in the original paper is the approach that was taken to weight initialization. For all of the results presented below, the bias weights are initialized to .1 and all other initial weight values are drawn from $\mathcal{N}(0, .0001)$.

3 Results

Results are reported below for three of the seven games mentioned in the original paper: Breakout, Seaquest and Enduro. Successful policies are learned for each game. Table 1 compares the best average reward received for each game with the corresponding results from [6].

Figure 1 illustrates the impact of hyper-parameter selection on the success of the deep Q-learning algorithm. Several trends are apparent in the data. First, it appears that the larger value of ρ tends to lead to better learning results, particularly for Enduro and Seaquest. Although the results are not presented here, a number of unsuccessful preliminary tests were performed on Breakout with $\rho = .9$.

Second, at least among values examined here, it appears that no single set of hyper-parameters is optimal across all games. For example, the best hyper-parameter settings for Enduro results in no learning for Seaquest. It is possible to find settings that work reasonably well across multiple games, but any one choice will be a compromise.

[1] `https://github.com/spragunr/deep_q_rl`

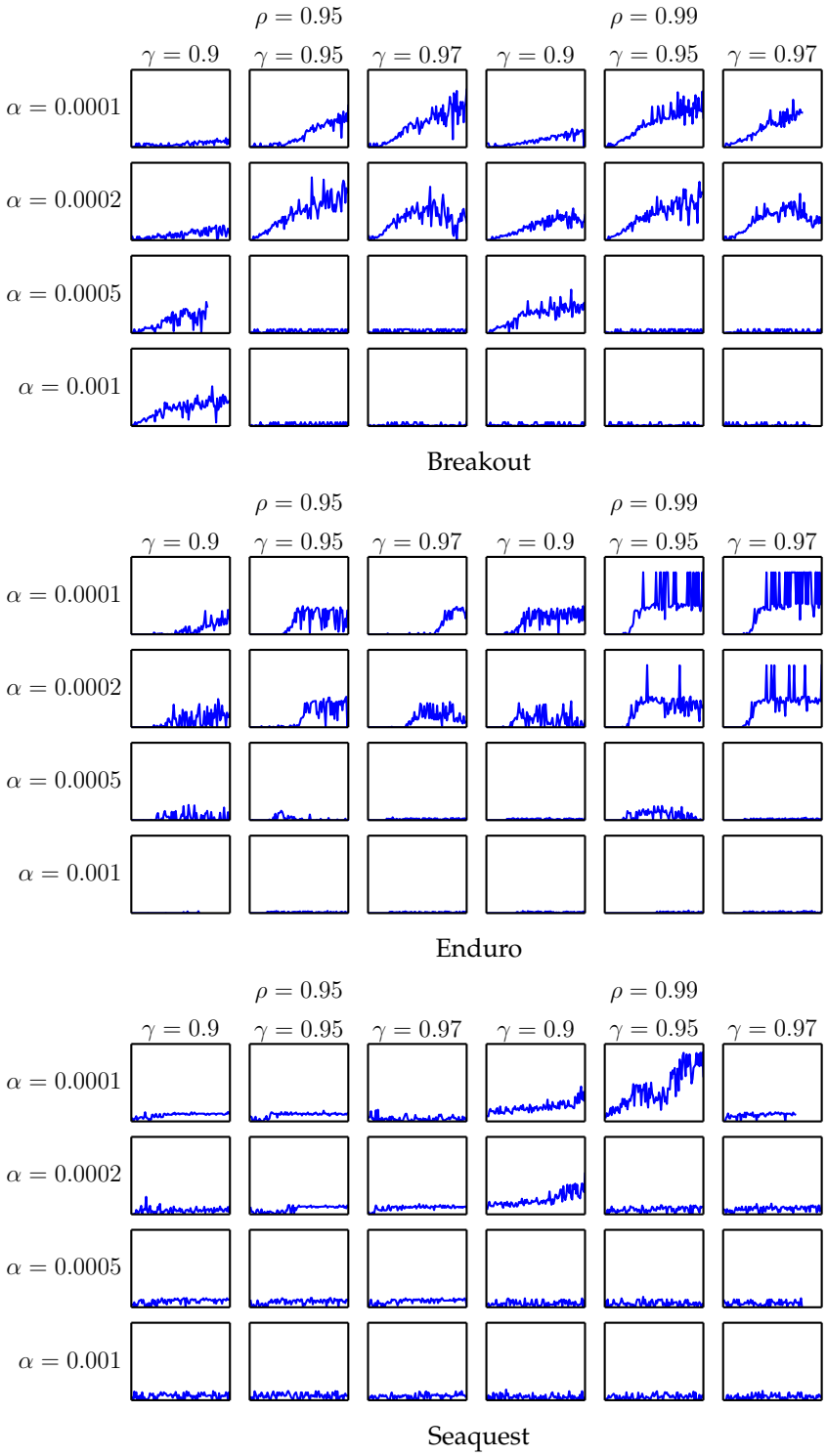


Figure 1: Each graph represents one training run with the indicated parameter settings for the indicated game. Training takes place over 100 epochs where each epoch represents 50,000 actions. Learning is evaluated after every epoch by testing the learned policy for 10,000 steps with an ϵ-greedy policy using $\epsilon = .05$. The average per-game reward is plotted for each epoch. The lower bounds for all y-axes are 0. The upper bounds are 200 for Breakout, 1000 for Enduro, and 2500 for Seaquest. (Note that the incomplete data in some of the graphs represent jobs that had not completed at the time of submission.)

The most striking observation is that some games show significantly more sensitivity to hyper-parameter selection than others. In particular, 21 of the 24 different settings for Seaquest show no noticeable learning progress. Only one setting results in learning that is comparable to reported results.

	Breakout	Enduro	Seaquest
DQN [6]	**168**	470	1705
DQN (current results)	162	**804**	**2228**

Table 1: Maximum average total reward for learned policies. The results labeled "DQN [6]" represent the best policies discovered using an unspecified, but constant, set of parameters. The results labeled "DQN (current results)" represent the maximum values across all of the graphs in Figure 1.

Note that the graphs in Figure 1 represent a single training run for each parameter setting. It has been our experience that learning results tend to be reasonably consistent for a given set of hyper-parameters, but further experiments would be necessary to quantitatively determine how hyper-parameter values impact the variability of learning.

4 Conclusion

Even in the realm of supervised learning, the problem of tuning hyper-parameters for gradient-based optimization of neural networks is notoriously difficult. The situation becomes worse when these networks are used in conjunction with value function estimation. In supervised learning progress can be tracked by observing the value of the loss function or the error on a validation set. In reinforcement learning the value of the loss function is not a reliable indicator of progress: the value function is a moving target, so increases in the loss function may represent changes in the magnitude of the estimated value function rather than a lack of progress in learning. The only reliable way to monitor progress is to periodically evaluate the learned policy. This process is noisy, slow, and computationally expensive.

Given these challenges, it is particularly desirable that a reinforcement learning algorithm that incorporates deep neural networks be relatively robust to hyper-parameter selection. The results reported here suggest that improving the robustness and reliability of deep Q-learning may be a valuable avenue for future research.

These results were prepared in reference to the original deep Q-learning paper [6]. The authors of that paper have recently published an extended set of results obtained using a slightly modified version of the algorithm [7]. The updated version of the algorithm periodically copies the network weights so that the target Q-values are calculated using weights that are held constant across many updates. The authors report that this modification improves the stability of the algorithm. It will be a focus of future work to determine how this modification impacts the algorithm's sensitivity to hyper-parameter selection.

5 Acknowledgments

An initial version of the code described here was posted on-line in September 2014. A number individuals have contributed suggestions and improvements since then. I am particularly indebted to Alejandro Dubrovsky for his contributions.

References

[1] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 06 2013.

[3] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[4] Justin Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, pages 369–376, 1995.

[5] Sander Dieleman. My solution for the galaxy zoo challenge. `http://benanne.github.io/2014/04/05/galaxy-zoo.html`, 2014.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. 2013.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis

Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[8] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*. Citeseer, 1993.

[9] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Reinforcement learning objectives constrain the cognitive map

Kimberly L. Stachenfeld
Princeton Neuroscience Institute
Princeton University
Princeton, NJ 08544
kls4@princeton.edu

Matthew M. Botvinick
Psychology Department & Princeton Neuroscience Institute
Princeton University
matthewb@princeton.edu

Samuel J. Gershman
Department of Brain & Cognitive Sciences
Massachusetts Institute of Technology
sjgershm@mit.edu

Abstract

In this work, we detail a model of the cognitive map predicated on the assumption that spatial representations are optimized for maximizing reward in spatial tasks. We describe how this model gives rise to a number of experimentally observed behavioral and neural phenomena, including neuronal populations known as place and grid cells. Place and grid cells are spatially receptive cells found in the hippocampus and entorhinal cortex, respectively. Classic place cells have a single firing field tied to a specific location in space. The firing properties of these cells are sensitive to behaviorally relevant conditions in the environment; for instance, they tend to be skewed along commonly traveled directions, clustered around rewarded locations, and influenced by the geometric structure of the environment. Grid cells exhibit multiple firing fields arranged periodically over space. These cells reside in the entorhinal cortex, and vary systematically in their scale, phase, and orientation.

We hypothesize that place fields encode not just information about the current location, but also predictions about future locations under the current policy. Under this model, a variety of place field phenomena arise naturally from the disposition of rewards and barriers and from directional biases as reflected in the transition policy. Furthermore, we demonstrate that this representation of space can support efficient reinforcement learning (RL). We also propose that grid cells compute the eigendecomposition of place fields, one result of which is the segmentation of an enclosure along natural boundaries. When applied recursively, this segmentation can be used to discover a hierarchical decomposition of space, allowing grid cells to support the identification of subgoals for hierarchical RL. This suggests a substrate for the long-standing finding that humans tend to divide space hierarchically, resulting in systematic biases about relations between locations in different regions.

Keywords: spatial navigation, hippocampus, reinforcement learning

Acknowledgements

We are very grateful for the funding provided for this project by the NSF Collaborative Research in Computational Neuroscience (CRCNS) Program Grant IIS-120 7833 and The John Templeton Foundation. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

1 Introduction

Traditionally, efforts to understand the cognitive map have sought to characterize the receptive fields of spatial cells in the hippocampus and entorhinal cortex and explain what neural computations could be responsible for the implementation. Our work has been motivated instead by considering the downstream purpose of these representations. The question we hope to address is to what extent place and grid cell encoding can be understood as an optimal substrate for maximizing expected future reward.

We consider first the problem of reward optimization in a Markov decision process, which consists of the following: a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition distribution $P(s'|s,a)$ specifying the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ after taking action $a \in \mathcal{A}$, a reward function $R(s)$ specifying the expected reward in state s, and a discount factor $\gamma \in [0,1]$. An agent chooses actions according to a policy $\pi(a|s)$ and collects rewards as it moves through the state space. The standard RL problem is to choose a policy that maximizes the *value* (expected discounted future return), $V(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0 = s\right]$. In our simulations we feed the agent the optimal policy; however, the policy dependent computations considered can be nested within algorithms for policy improvement. To simplify notation, we assume implicit dependence on π and define the state transition matrix T, where $T(s,s') = \sum_a \pi(a|s)P(s'|s,a)$.

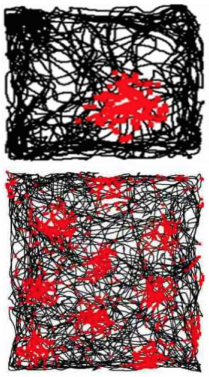


Figure 1: Sample place field (above) and grid field (below)

Most work on RL has focused on model-free and model-based algorithms [2]. However, there exists a third class that has received less attention. As shown by Dayan [3], the value function can be decomposed into the inner product of the reward function with the "successor representation," denoted by M:

$$V(s) = \sum_{s'} M(s,s')R(s'), \qquad M = (I - \gamma T)^{-1} \tag{1}$$

where I denotes the identity matrix. The SR encodes the expected discounted future occupancy of state s' along a trajectory initiated in state s and can be expressed as $M(s,s') = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}\{s_t = s'\} \mid s_0 = s\right]$, $\mathbb{I}\{\cdot\} = 1$ if its argument is true, and 0 otherwise.

The SR obeys a recursion analogous to the Bellman equation for value functions:

$$M(s,j) = \mathbb{I}\{s = j\} + \gamma \sum_{s'} T(s,s')M(s',j). \tag{2}$$

This recursion can be harnessed to derive a temporal difference learning algorithm for incrementally updating an estimate $\hat{M}$ of the SR [3, 8]. After observing a transition $s \to s'$, the estimate is updated according to:

$$\hat{M}(s,j) \leftarrow \hat{M}(s,j) + \eta\left[\mathbb{I}\{s = j\} + \gamma\hat{M}(s',j) - \hat{M}(s,j)\right], \tag{3}$$

where η is a learning rate (unless specified otherwise, $\eta = 0.1$ in our simulations). The SR represents an elegant compromise between model-free and model-based algorithms: like model-free algorithms, policy evaluation is computationally efficient, but at the same time the SR provides some of the same flexibility as model-based algorithms.

2 The successor representation and place cells

In this section, we explore the neural implications of using the SR for policy evaluation: if the brain encoded the SR, what would the receptive fields of the encoding population look like, and what would the population look like at any point in time? This question is most easily addressed in spatial domains, where states index spatial locations. For an open field with uniformly distributed rewards we assume a random walk policy, and the resulting SR for a particular location is an approximately symmetric, gradually decaying halo around that location (Fig. 2a)—the canonical description of a hippocampal place cell. In order for the population to encode the expected visitations to each state in the domain from the current starting state (i.e. a row of M), each receptive field corresponds to a column of the SR matrix. This allows the current state's value to be computed by taking the dot product of its population vector with the reward vector. The receptive field (i.e. column of M) will encode the discounted expected number of times that state was visited for each starting state, and will therefore skew in the direction of the states that likely preceded the current state.

For this reason, when an animal has been trained to travel in a preferred direction along a linear track, we expect place fields to become skewed opposite the direction of travel as has been observed experimentally [16, 17]. When barriers are inserted into the environment, the probability of transitioning across these obstacles will go to zero such that receptive fields will be discontinuous across barriers (Fig. 2c,e). In this way SR place fields are constrained by environmental geometry. Consistent with this idea, experiments have shown that place fields become distorted around barriers (Fig. 2h) [20, 26].

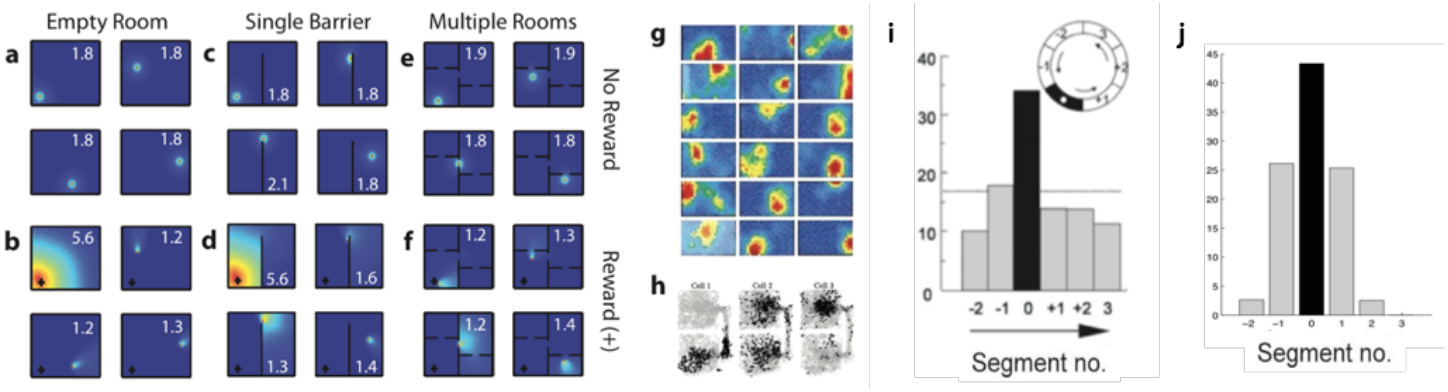


Figure 2: **SR place fields**. (a–f) Simulated SR place fields. Top two rows are unrewarded conditions, bottom two show place fields near reward (marked by +). Maximum firing rate (a.u.) indicated for each plot. (a, b) Empty room. (c, d) Single barrier. (e, f) Multiple rooms. (g–h) Experimentally recorded place fields from rats in (g) an empty room [30] and (h) rooms with barrier [26]. (i–j) Percentage of neurons firing in segments of an annular water maze for (i) experimental hippocampal recordings [12] and (j) simulated SR place fields.

Another way to alter the transition policy is to introduce a goal, which induces a tendency to move in the direction that maximizes reward. Under these conditions, we expect firing fields centered near rewarded locations to expand to include the surrounding locations and to increase their firing rate (Fig. 2b,d,f), as has been observed experimentally [6]. Meanwhile, we expect the majority of place fields that encode non-rewarded states to skew slightly away from the reward (Fig. 2b,d,f). Under certain parameter settings, the spread of the rewarded locations' fields compensates for the skew of surrounding fields away from the reward, and we observe "clustering" around rewarded locations, as has been observed experimentally in the annular water maze task (Fig. 2i,j) [12]. However, this parameterization sensitivity may explain why goal-related firing is not observed in all tasks [13].

In related previous work, Gustafson and Daw [9] showed how topologically-sensitive spatial representations recapitulate many aspects of place cells and grid cells that are difficult to reconcile with a purely Euclidean representation of space. They also showed how encoding topological structure greatly aids reinforcement learning in complex spatial environments. Our contribution was to show that the SR naturally encodes topological structure in a format that enables efficient RL.

3 Eigendecomposition of the successor representation: hierarchical decomposition and grid cells

Reinforcement learning and navigation can often be made more efficient by decomposing the environment hierarchically. For example, the options framework [29] utilizes a set of subgoals to divide and conquer a complex learning environment. Recent experimental work suggests that the brain may exploit a similar strategy [1, 22, 5]. A key problem, however, is discovering useful subgoals; while progress has been made on this problem in machine learning, we still know very little about how the brain solves it (but see [23]). In this section, we show how the eigendecomposition of the SR can be used to discover subgoals. The resulting eigenvectors strikingly resemble grid cells observed in entorhinal cortex.

A number of authors have used graph partitioning techniques to discover subgoals [18, 25]. These approaches cluster states according to their community membership (a community is defined as a highly interconnected set of nodes with relatively few outgoing edges). Transition points between communities (bottleneck states) are then used as subgoals. One important graph partitioning technique, used by [25] to find subgoals, is the normalized cuts algorithm [24], which recursively thresholds the eigenvector with the second smallest eigenvalue (the Fiedler vector) of the normalized graph Laplacian to obtain a graph partition. Given an undirected graph with symmetric weight matrix W, the graph Laplacian is given by $L = D - W$. The normalized graph Laplacian is given by $\mathcal{L} = I - D^{-1/2}WD^{-1/2}$, where D is a diagonal degree matrix with $D(s,s) = \sum_{s'} W(s,s')$. When locations on the graph are projected onto the eigenvectors of L, they cluster according to their community membership (with the Fiedler vector clustering points according to the two broadest communities). Locations in distinct communities but close in Euclidean distance – for instance, nearby points on opposite sides of a boundary – will therefore be represented as distant in the eigenspace.

The normalized graph Laplacian is closely related to the SR [14]. Under a random walk policy, the transition matrix is given by $T = D^{-1}W$. If ϕ is an eigenvector of the random walk's graph Laplacian $I - T$, then $D^{1/2}\phi$ is an eigenvector of the normalized graph Laplacian. The corresponding eigenvector for the discounted Laplacian, $I - \gamma T$, is $\gamma\phi$. Since

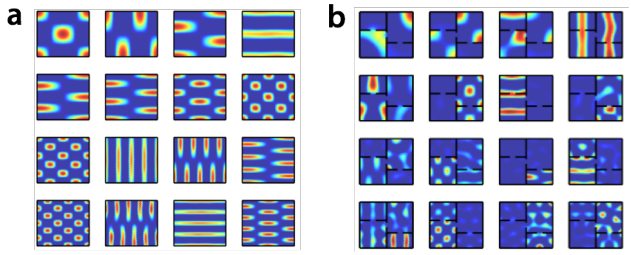


Figure 3: **Thresholded eigendecomposition of the SR**. Each panel shows the same eigenvectors sampled from the top 100 (excluding the constant first eigenvector) for two of the environmental geometries shown in Fig. 2 (no reward). (a) Empty room. (b) Single barrier. Thresholded at zero to eliminate negative firing rates.

the matrix inverse preserves the eigenvectors, the normalized graph Laplacian has the same eigenvectors as the SR, $M = (I - \gamma T)^{-1}$, scaled by $\gamma D^{-1/2}$.

A number of representative SR eigenvectors are shown in Fig. 3 for two different room topologies. The higher frequency eigenvectors display the latticing characteristic of grid cells [10]. The eigendecomposition is often discontinuous at barriers, and in many cases different rooms are represented by non-overlapping sinusoids. Importantly, the grid fields show compartmentalization resembling that observed experimentally by Derdikman and Moser [4].

In the multiple rooms environment, visual inspection confirms that the SR eigenvector with the second smallest eigenvalue divides the enclosure along the vertical barrier: the left half is almost entirely blue and the right half almost entirely red, with a smooth but steep transition at the doorway (Fig. 4a). Applying this segmentation by thresholding recursively, as in the normalized cuts algorithm, produces a hierarchical decomposition of the environment (Fig. 4b,c). By identifying useful subgoals from the environmental topology, this decomposition can be exploited by hierarchical learning algorithms [1, 23].

4 Further discussion of hierarchical maps

In 1978, Stevens and Coupe [28] reported behavioral evidence that people represent space hierarchically. Human subjects would, for instance, systematically conclude that San Diego, CA is further west than Reno, NV, despite this being untrue. This can be explained by considering the spectral representation of these locations in the entorhinal cortex. Different nodes that lie in the same community map to nearby locations in the space of the eigenvectors, a result that justifies the commonly used technique of "spectral clustering." Furthermore, if we apply the recursive thresholding and segmentation discussed in the previous section, two locations in the same community will map to the exact same thresholded value. A point in this thresholded eigenspace is therefore represented as a state abstraction, with different states mapped to a single point. It is straightforward to imagine how information learned about points in this space–such as which is west of the other–necessarily generalize to all states that map to the same (or perhaps nearby) spectral coordinates. Thus, you expect to see these hierarchically organized generalizations.

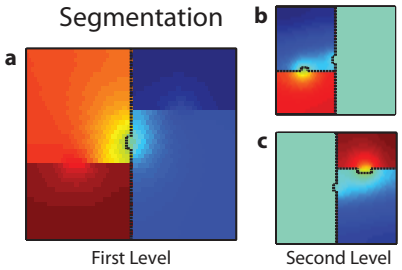


Figure 4: **Normalized cut segmentation**. (a) The results of segmentation by thresholding the second eigenvector of the multiple rooms environment in Fig. 2. Dotted lines indicate the segment boundaries. (b, c) Eigenvector segmentation applied recursively to fully parse the enclosure into the four rooms.

Furthermore, Stevens and Coupe [28] showed that people overestimate the distance between two locations when they were separated by a boundary (e.g., a state or country line), again hypothesizing this to arise from a hierarchical organization of space (see also [11]). Under our model (using the difference in activation for states given initial state s, $|M(s,s) - M(s,s')|$, as a proxy for the perceived distance between s and s'), distance estimates between points in different regions of the environment are altered when an enclosure is divided by the weak boundary that segmentation would effectively impose. We see that as the permeability of the barrier decreases (making the boundary harder to cross), the percent increase in perceived distance between locations increases without bound. This gives rise to a discontinuity in perceived travel time at the weak boundary. Interestingly, the hippocampus is directly involved in distance estimation in large-scale environments [19], suggesting the hippocampal cognitive map as a neural substrate for distance biases.

5 Conjunctivity and repeated firing fields

In addition to the previously discussed phenomena, place cells can develop conjunctivity and often exhibit multiple firing fields. Conjunctivity refers to joint selectivity for multiple stimulus dimensions, such as both location *and* smell at that location or time during the trial, and tends to arise predominantly when aspects of the stimulus are jointly relevant to the task [15]. Additional firing fields often arise for the same place cell in different rooms, but at the same corresponding

location relative to the boundaries of the new room [27]. If the dimensions of the new room are changed, the place fields will even "stretch" to maintain their position relative to boundaries [21].

To address these phenomena, we note that it is non-trivial for the animal to compute location from the stream of sensory input it experiences, and that "locations" in the cognitive map should be thought of as latent variables. We hypothesize that perhaps the cognitive map is conservative in assigning new variables when not necessary. Under this interpretation, a place cell with multiple firing fields is just representing two different but similar locations as the same. Similarly, a place cell might learn conjunctivity as the animal learns that two distinct conditions occur in the same location. That is, the place field still has only the one and the same receptive field in *task space*, but task space no longer has a one-to-one mapping onto its spatial correlates.

To realize this intuition, we have begun to fit a sticky Hierarchical Dirichlet Process - Hidden Markov Model [7]. This is a Bayesian non-parametric model in which observations at each location are generated by a Markov process over latent states, and a new state will be inferred if it is sufficiently improbable that a previously observed state is responsible. "Hierarchical" refers to the fact that the model allows for switching between "contexts" in which different Markov chains are active. This indirectly encodes global remapping: if context changes suddenly and dramatically, it no longer becomes plausible that the same location will be encountered, and it becomes useful to draw upon a novel map.

References

[1] M. Botvinick, Y. Niv, and A. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113:262–280, 2009.

[2] N. Daw, Y. Niv, and P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8:1704–1711, 2005.

[3] P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5:613–624, 1993.

[4] D. Derdikman, J. R. Whitlock, A. Tsao, M. Fyhn, T. Hafting, M.-B. Moser, and E. I. Moser. Fragmentation of grid cell maps in a multicompartment environment. *Nature Neuroscience*, 12:1325–1332, 2009.

[5] C. Diuk, K. Tsai, J. Wallis, M. Botvinick, and Y. Niv. Hierarchical learning induces two simultaneous, but separable, prediction errors in human basal ganglia. *The Journal of Neuroscience*, 33:5797–5805, 2013.

[6] A. Fenton, L. Zinyuk, and J. Bures. Place cell discharge along search and goal-directed trajectories. *European Journal of Neuroscience*, 12:3450, 2001.

[7] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. An hdp-hmm for systems with state persistence. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 312–319, New York, NY, USA, 2008. ACM.

[8] S. Gershman, C. Moore, M. Todd, K. Norman, and P. Sederberg. The successor representation and temporal context. *Neural Computation*, 24:1553–1568, 2012.

[9] N. J. Gustafson and N. D. Daw. Grid cells, place cells, and geodesic generalization for spatial reinforcement learning. *PLoS Computational Biology*, 7:e1002235, 2011.

[10] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436:801–806, 2005.

[11] S. C. Hirtle and J. Jonides. Evidence of hierarchies in cognitive maps. *Memory & Cognition*, 13:208–217, 1985.

[12] S. A. Hollup, S. Molden, J. G. Donnett, M. B. Moser, and E. I. Moser. Accumulation of hippocampal place fields at the goal location in an annular watermaze task. *Journal of Neuroscience*, 21:1635–1644, 2001.

[13] P. Lenck-Santini, E. Save, and B. Poucet. Place-cell firing does not depend on the direction of turn in a y-maze alternation task. *European Journal of Neuroscience*, 13(5):1055–8, 2001.

[14] S. Mahadevan. Learning representation and control in markov decision processes: New frontiers. *Foundations and Trends in Machine Learning*, 1:403–565, 2009.

[15] S. McKenzie, A. Frank, N. Kinsky, B. Porter, P. Riviere, and H. Eichenbaum. Hippocampal representation of related and opposing memories develop within distinct, hierarchically organized neural schemas. *Neuron*, 83:202–215, 2014.

[16] M. R. Mehta, C. A. Barnes, and B. L. McNaughton. Experience-dependent, asymmetric expansion of hippocampal place fields. *Proceedings of the National Academy of Sciences*, 94:8918–8921, 1997.

[17] M. R. Mehta, M. C. Quirk, and M. A. Wilson. Experience-dependent asymmetric shape of hippocampal receptive fields. *Neuron*, 25:707–715, 2000.

[18] I. Menache, S. Mannor, and N. Shimkin. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning*, pages 295–306. Springer, 2002.

[19] L. K. Morgan, S. P. MacEvoy, G. K. Aguirre, and R. A. Epstein. Distances between real-world locations are represented in the human hippocampus. *The Journal of Neuroscience*, 31:1238–1245, 2011.

[20] R. U. Muller and J. L. Kubie. The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells. *The Journal of Neuroscience*, 7:1951–1968, 1987.

[21] J. O'Keefe and N. Burgess. Geometric determinants of the place fields of hippocampal neurons. *Nature*, 381:425–8, 1996.

[22] J. J. Ribas-Fernandes, A. Solway, C. Diuk, J. T. McGuire, A. G. Barto, Y. Niv, and M. M. Botvinick. A neural signature of hierarchical reinforcement learning. *Neuron*, 71:370–379, 2011.

[23] A. C. Schapiro, T. T. Rogers, N. I. Cordova, N. B. Turk-Browne, and M. M. Botvinick. Neural representations of events arise from temporal community structure. *Nature Neuroscience*, 16:486492, 2013.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22:888–905, 2000.

[25] Ö. Şimşek, A. P. Wolfe, and A. G. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 816–823. ACM, 2005.

[26] W. Skaggs and B. McNaughton. Spatial firing properties of hippocampal ca1 populations in an environment containing two visually identical regions. *Journal of Neuroscience*, 18:8455–8466, 1998.

[27] H. Spiers, R. Hayman, A. Jovalekic, E. Marozzi, and K. Jeffery. Place field repetition and purely local remapping in a multicompartment environment. *Cerebral Cortex*, pages 10–25, 2015.

[28] A. Stevens and P. Coupe. Distortions in judged spatial relations. *Cognitive Psychology*, 10:422 – 437, 1978.

[29] R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[30] M. Wilsion and B. McNaughton. Dynamics of the hippocampal ensemble code for space. *Science*, 261(5124):1055–1058, 1993.

The successor representation in human reinforcement learning: evidence from retrospective revaluation

Ida Momennejad
Princeton Neuroscience Institute
Princeton University
idam@princeton.edu

Jin H. Cheong
Princeton Neuroscience Institute
Princeton University
jcheong@princeton.edu

Matthew M. Botvinick
Princeton Neuroscience Institute
Princeton University
mattewb@princeton.edu

Samuel J. Gershman
Department of Brain and Cognitive Sciences, MIT
sjgershm@mit.edu

Abstract

Reinforcement learning (RL) has been posed as a competition between model-free (MF) and model-based (MB) learning. MF learning cannot solve problems such as revaluation or latent learning, hallmarks of MB behavior. However, we suggest that varieties of MB-like behavior are equally well or better predicted by a third solution to the RL problem: the successor representation (SR). We conducted two experiments to test this hypothesis, comparing the classic 'reward devaluation' (reward structure changes, transition structure stays the same) with 'transition devaluation' (reward structure stays the same, transition structure changes). Behaviorally, we found that while subjects were sensitive to both conditions, they were more sensitive to reward than transition devaluation. Furthermore, faster responses showed greater sensitivity to reward than transition devaluation, while slower responses displayed equal sensitivity to both. MF, MB, and mixture models do not predict this asymmetry between reward and transition devaluation. On the other hand, a pure SR strategy will only be sensitive to reward but not transition devaluation, because the successor representation effectively "compiles" the transition structure and therefore cannot adapt quickly to changes. We propose two novel SR mixture models that can better explain our asymmetrical findings. One model is an extension of the Dyna style architecture, SR-Dyna, in which the successor representation is computed during real experience and updated via simulated experience during periods of episodic replay and pre-play. Another is a mixture SR-MB strategy, whereby the value function for a model-based strategy is initialized using the SR. While SR-Dyna combines the advantages of incremental learning with planning, SR-MB may be better suited to decision making under time-pressure. Compared to MB and MF RL methods, the successor representation (SR) family of models better captures the asymmetry in our behavioral findings.

Keywords: Successor representation, Retrospective revaluation, Reinforcement learning, Human behavior, Dyna architecture

Acknowledgements

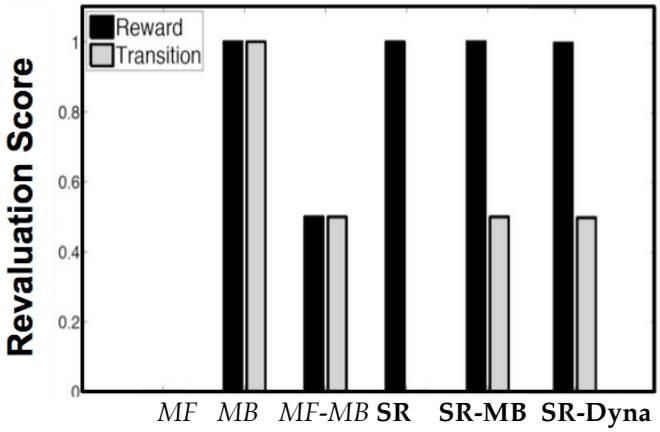
This project was made possible through grant support from the National Science Foundation (CRCNS 1207833) and the John Templeton Foundation. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

1 Introduction

When faced with a reinforcement-learning (RL) problem, model-free (MF) learning uses cached values to determine decisions. Model-based (MB) learning, on the other hand, uses a model of transitions and rewards to determine the optimal policy. Many important RL problems, such as retrospective revaluation or latent learning, cannot be solved by a purely model-free strategy and are considered hallmarks of model-based learning (or a mixture of MF-MB). However, MB strategies are working memory intensive, error-prone, and even intractable for large search spaces such as wide and deep trees (Lengyel and Dayan, 2008; Collins et al., 2012). At the same time, varieties of MB-like behavior are widely prevalent across animals. This prevalence of MB-like behavior together with the overwhelming computational cost and error-proneness of MB strategies may suggest that less computationally costly mechanisms are at play. Here we tested the hypothesis that human solutions to classical MB problems such as retrospective revaluation can be simulated by a third strategy that lies between model-free and model-based learning: the successor representation (SR; Dayan, 1993; Gershman et al., 2012; Stachenfeld et al., 2014). We suggest that a family of models including the successor representation strategy can better explain observed differences under reward vs. transition devaluation, as we explain below.

To solve the RL problem, the SR (*M* in Figure 1.B. notation) represents each state by the expected discounted future occupancy of its successor states. The value function can in turn be computed by taking the inner product of the SR and the reward function. The SR approach to value computation has two advantages. First, it renders value computation a linear operation, which is both biologically plausible and comparable in computational complexity to the MF system. Second, the SR (like the MB system) is sensitive to reinforcer devaluation, but with lower computational cost. Thus, SR lies between MF and MB strategies in terms of computational demand and sensitivity to devaluation. Here we also suggest that a biologically plausible mechanism for episodic consolidation and replay could update the SR via simulated experience, in absence of direct experience. This hybrid Dyna-like solution (see below) is sensitive to transition devaluation as well.

A. Comparison of model solutions to devaluation



B. Value function and the successor representation (M)

$$V(s) = \sum_{s'} M(s,s')R(s') \quad M = (I - \gamma T)^{-1}$$

C. SR-Dyna architecture

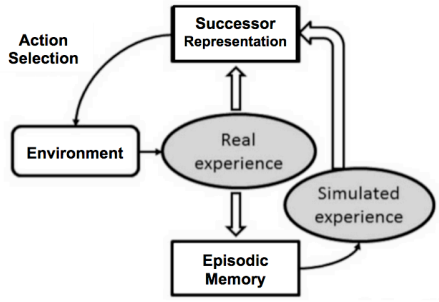


Figure 1. *(A) Model predictions for reward and transition revaluation. Predicted revaluation scores for model-free (MF), model based (MB), mixture model (MB-MF), purely successor representation (SR), hybrid SR-MB, and the SR-Dyna architecture for reward (black) vs. transition (gray) devaluation conditions. Only the SR family of models makes asymmetrical predictions for reward and transition devaluation (as shown in the following sections, data points at the alternative SR-MB or SR-Dyna models, fit to data in figure 4). (B) Value function for state s is computed as the inner product of the successor representation M and the reward vector R. The successor representation is computed using the inverse of the discounted transition matrix (gamma is the discount factor, over the expected utility of the frequency of visit to a successor state). (C) The SR-Dyna architecture. SR-Dyna incorporates the successor representation, learned*

via real experience, and an episodic reinstatement mechanism that updates the SR via replay and preplay (simulated experience).

In 2 experiments, we asked whether humans behave in accordance with the SR. We compared solutions to the classic 'reward devaluation' (reward structure changes, transition structure stays the same) with 'transition devaluation' (reward structure stays the same, transition structure changes) among models and in human behavior. MB and MF solutions predict symmetrical responses to the problem: MF predicts no revaluation to either devaluation condition, while the MB and mixture MF-MB models predict equal revaluation under both types of devaluation (Figure 1). A pure SR strategy, on the other hand, predicts an asymmetrical response as it is only sensitive to reward devaluation but not sensitive to transition devaluation, because on its own SR does not update itself in absence of direct experience. The reason for this asymmetry is that the SR effectively "compiles" the transition structure and therefore cannot adapt quickly to changes. That said, a family of hybrid models offer a possible solution to this shortcoming where the SR initializes a MB strategy that solves transition devaluation; or more interestingly, a Dyna-like solution updates the SR via replay (Sutton, 1990). Dyna-like architectures treat planning as a learning problem that is enhanced via simulated or imagined experience. We thus suggest SR-Dyna, an extension of the Dyna architecture, where the SR is updated via virtual experience generated by a simulation mechanism, e.g. episodic reinstatement or sampling. Notably, SR-Dyna solves MB-like problems in absence of MB computation.

2 Methods

2.1 Task

We designed a task to study revaluation behavior under varieties of devaluation. The schematic experimental conditions are shown as Markov reward processes (MRPs; i.e., Markov decision processes without the decision component) in Figure 1. States are represented as numbered circles and arrows specify transitions. Participant passively experienced these transitions throughout the experiment while performing a cover task. Every MRP consisted of 3 states; each tagged with a distinct background color in the experiment and contained a distinct image of a face, scene, or an object. Each MRP consisted of 6 states with state transitions that divided them into two trajectories. In the first phase of the experiment, participants passively observed the transition among states while performing a category judgment on the images associated in each state (face, scene, object).

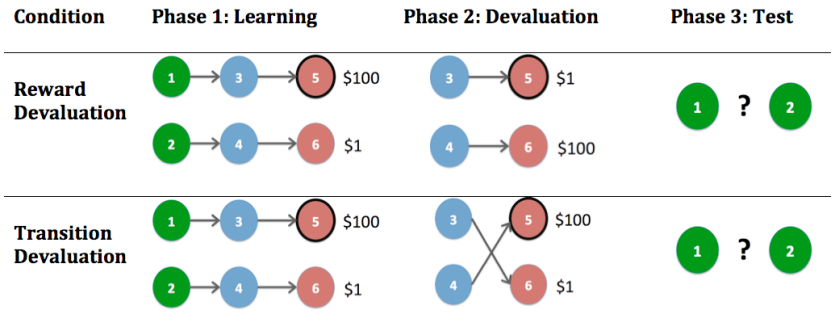


Figure 2. Schematic design
Revaluation scores are calculated as the difference between the subject's preference between state 1 and 2 during phase 3 (following devaluation) and their preference after phase 1 (learning).

We conducted 2 studies that were identical except for the preference rating condition (Figure 3). In study 1, participants had virtually unlimited time (60 s) to rate their preference between states 1 and 2 on a sliding scale. In study 2, participants made a forced choice between the two starting states with a 1.5 s deadline. Both studies consisted of 20 trials (or games), each corresponding to one of three conditions: reward devaluation (8 trials), transition devaluation (8 trials), and catch trials or the control condition (4 trials).

During phase 1, participants passively experienced all states and their associated reward. The passive presentation of stimuli and trajectories enabled us to control exposure to the entire state space across participants. To ensure that participants attended to each state and assess the extent to which they were attending to the stimuli, they were asked to perform a category judgment task during the passive navigation of the MRPs. Phase 1 was concluded once the participant reached learning criterion, or after 20 stimulus presentations. Trials in which participants did not learn the trajectories leading to most rewarding states were excluded from further analysis.

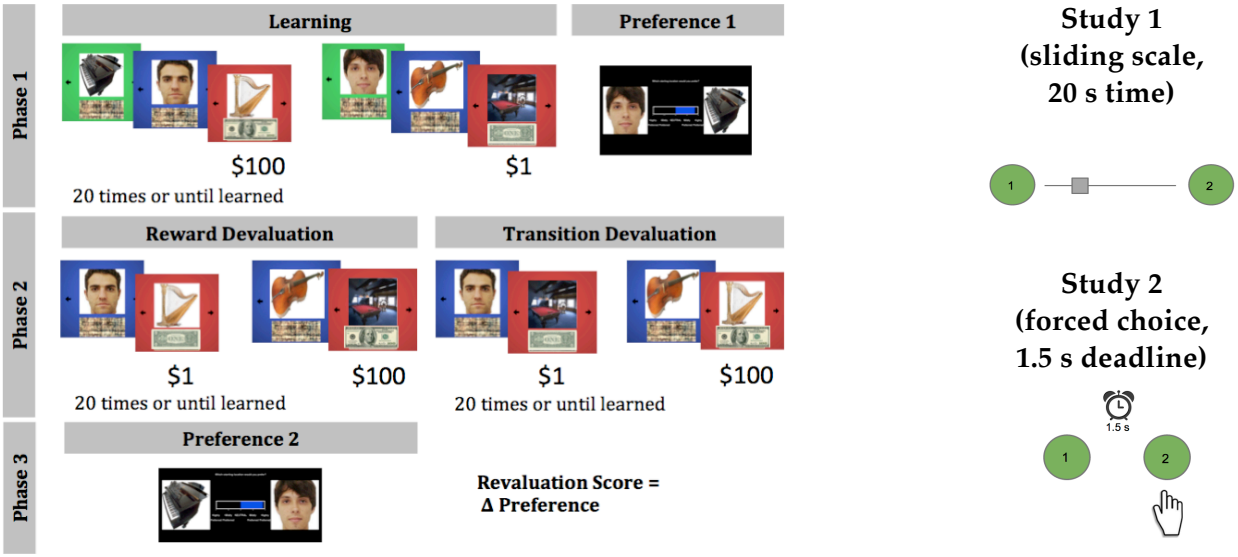


Figure 3. *(Left) The time course of a trial in the experimental paradigm. (Right) In Study 1 participants used a sliding scale with virtually no deadline (20 s delay) in order to rate their preference. In Study 2, preference rating after both phases was a forced choice with a short deadline (1.5 s).*

During Phase 2, participants passively viewed all states except the starting states of each trajectory (states 1 and 2 in Figure 1, see also Phase 2 in Figure 2). As in Phase 1, participants performed a category judgment on the images of the states they visited. This category task served as a measure of attention to the states during both Phase 1 and Phase 2. During Phase 2, depending on the trial condition, any of the following three changes took place. In the reward devaluation condition, the rewards associated with end states changed. In the transition devaluation condition, the transition structure between second and third states within each trajectory changed (Figures 1 and 2). In the control condition (no devaluation) no changes took place in the second phase. The last condition controlled for participants' choices in absence of any devaluation, ensuring that any differences in starting state preferences between Phase 1 and Phase 3 cannot be explained by simple forgetting of the starting states (memory decay) or by making more random responses.

3 Results

3.2 Study 1 (n = 58) results

69 participants were recruited for Study 1, 4 participants were excluded as they did not learn the task and could not finish the study within the allotted 1.5 hours. 7 were removed from the final analysis due to accuracies below 80% in the categorization task, a threshold used as a measure of attention to /engagement with the experiment. Revaluation scores were computed as the difference in initial state preference between Phases 1 and 3 (Δ preference = initial state preference in Phase 3 – initial state preference in Phase 1). Revaluation score were computed separately for the reward devaluation and transition devaluation conditions.

The difference between the revaluation scores of the reward vs. transition devaluation conditions was

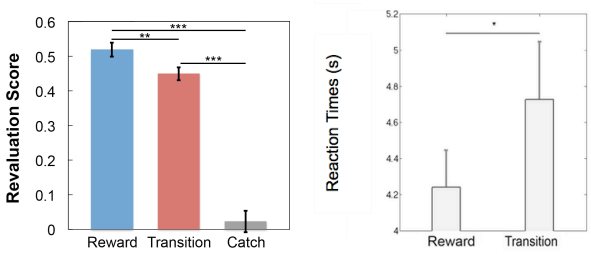


Figure 4. Study 1 results (n=58)
(Left) Revaluation behavior during reward devaluation, transition devaluation, and catch trials. (Right) reaction times to the preference decision under reward and transition devaluation ($p < .05$, ** $p < .005$, *** $p < .001$).*

significant at $p = .005$ (Figure 4). We also tested the conditions for any differences in reaction times to the reward or the transition revaluation, and found that responses to decisions under transition devaluation were significantly slower than decisions under reward devaluation ($p < .05$).

3.2 Study 2 (n= 52) results and model fits

60 participants were recruited for Study 2, 4 were excluded, as they did not learn the task. 4 were removed from the final analysis due to accuracies below 80% in the categorization task. The overall difference between the Study 2 revaluation scores for the reward devaluation vs. the transition devaluation conditions was not significant, but trending at $p = .0795$. Since the preferences were deadlined, we predicted that the difference between the conditions would be best elicited in the fastest responses, when participants had less time for computations and were more likely to rely on precomputed state structures. We thus divided the responses into four quartiles based on the reaction times (Figure 5). First, we found significant differences between revaluation scores under reward and transition devaluation in the first quartile, i.e. fastest responses ($p = .0208$). Our predictions also pertained to the difference between the fastest vs. the slowest responses. So we also tested for differences between revaluation scores between the first quartile (fastest responses) vs. the last quartile (slowest responses) and found that it was significant at $p = .0387$ (two-tailed).

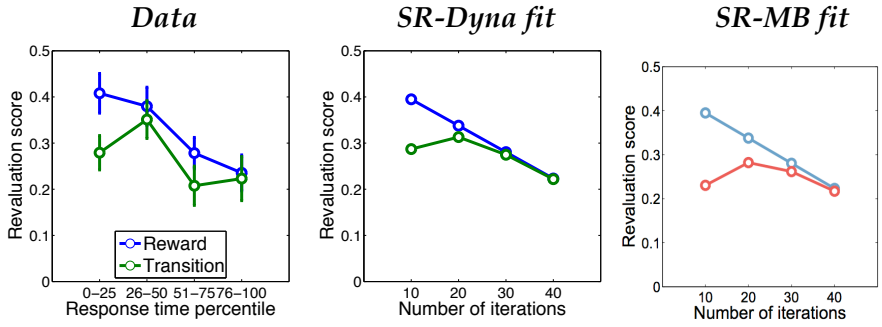


Figure 5. Study 2 Revaluation behavior under reward and transition devaluation conditions and model fits.
Left: Behavioral results are split in quartiles based on response times. For each subject we separated the trials into four quartiles on the basis of response times. In the 0-25th percentile RT's we observed a significant difference between reward and transition revaluation scores ($p = .0208$) and between revaluation scores in the first vs. the last percentile ($p = .0387$). Middle: Model fits for SR-Dyna ($r = .91$, $p = .002$). Right: Model fits for SR-MB ($r = .91$, $p = .002$).

We fit both hybrid SR models to the findings: SR-MB (Figure 5, right), and SR-Dyna, an extension of the Dyna-like architecture (Figure 1.C.). SR-MB assumes that participants carry out value iteration, initialized with the successor representation (SR). Both models assume that there is "computational noise" (due to attentional lapses or other cognitive errors) that increases linearly with the number of iterations. As such, revaluation magnitude is pulled towards 0. In other words, there is a tension between value iteration - which increases revaluation - and noise -which decreases it. This tension gives rise to the non-monotonic curves: with more processing time, the influence of the SR initialization diminishes (Figure 5). This can

be motivated by the idea that planning is error-prone and these errors compound, or that there was more noise in the slowest quartile as participants were simply distracted.

4 Discussion

We explored the hypothesis that humans use the successor representation to solve sequential decision problems. In two experiments we compared decisions in response to devaluation of rewards and contingencies in the transition structure (transition devaluation). We observed an asymmetrical pattern: decisions requiring retrospective revaluation of rewards were more accurate and faster than those requiring transition revaluation. The observed asymmetry of decisions under reward vs. transition devaluation is not predicted by classical RL solutions (i.e. MF, MB, MF-MB mixture). We suggest that a family of SR solutions can better account for the asymmetry in our empirical findings. It is possible that the brain uses either of two hybrid models: SR-MB, in which the SR initializes a "rough draft" that is subsequently refined by MB computation; or SR-Dyna, in which the SR is updated via virtual experience or episodic replays in a Dyna-like architecture. Crucially, our suggested SR-Dyna solution is free of any MB calculations. Future studies can test whether SR-Dyna can predict and simulate varieties of related neural findings such as forward and reverse replay or latent learning. Model fits with SR-MB and SR-Dyna were equally good. We thus suggest a novel addition to the Dyna family of solutions in reinforcement learning: SR-Dyna, a combined learning and planning strategy that uses the SR.

5 References

1. Botvinick, M. & Weinstein, A. (2014). Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society of London: Series B. 369,* 20130480.
2. Collins, A. G. E. & Frank, M. J. (2012). How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis. *Eur. J. Neurosci. 35, 1024–35.*
3. Daw, N. D., and Dayan, P. (2014) The algorithmic anatomy of model-based evaluation. *Philosophical transactions of the royal society, B 2014, 369,* 20130478.
4. Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation, 5,* 613–624.
5. Gershman, S.J., Moore, C.D., Todd, M.T., Norman, K.A., & Sederberg, P.B. (2012). The successor representation and temporal context. *Neural Computation, 24*, 1553–1568.
6. Lengyel, M., & Dayan, P. (2007). Hippocampal contributions to control: The third way. *NIPS 2007*.
7. Stachenfeld, K.L., Botvinick, M.M., & Gershman, S.J. (2014). Design principles of the hippocampal cognitive map. *Advances in Neural Information Processing Systems 27*.
8. Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning, 216–224.*

Policy Learning with Hypothesis based Local Action Selection

Bharath Sankaran*
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
bsankara@usc.edu

Jeannette Bohg
Max Planck Institute for Intelligent Systems
Tübingen, Germany 72076
jeannette.bohg@tuebingen.mpg.de

Nathan Ratliff
Max Planck Institute for Intelligent Systems
University of Stuttgart
nathan.ratliff@tuebingen.mpg.de

Stefan Schaal
University of Southern California
Max Planck Institute for Intelligent Systems
sschaal@usc.edu

Abstract

For robots to be effective in human environments, they should be capable of successful task execution in unstructured environments. Of these, many task oriented manipulation behaviors executed by robots rely on model based grasping strategies and model based strategies require accurate object detection and pose estimation. Both these tasks are hard in human environments, since human environments are plagued by partial observability and unknown objects. Given these difficulties, it becomes crucial for a robot to be able to operate effectively under partial observability in unrecognized environments. Manipulation in such environments is also particularly hard, since the robot needs to reason about the dynamics of how various objects of unknown or only partially known shape interact with each other under contact. Modelling the dynamic process of a cluttered scene during manipulation is hard even if all object models and poses were known. It becomes even harder to reasonably develop a process or observation model, with only partial information about the object class or shape. To enable a robot to effectively operate in *partially observable unknown environments* we introduce a policy learning framework where action selection is cast as a *probabilistic classification problem* on hypothesis sets generated from observations of the environment. The action classifier operates online with a global stopping criterion for successful task completion. The example we consider is object search in clutter, where we assume having access to a visual object detector, that directly populates the hypothesis set given the current observation. Thereby we can avoid the temporal modelling of the process of searching through clutter. We demonstrate our algorithm on two manipulation based object search scenarios; a *modified minesweeper simulation* and a *real world object search in clutter* using a dual arm manipulation platform.

Keywords: Hypothesis Classification, Greedy Action Selection, Policy Learning, Learning from Demonstration

Acknowledgements

This research was supported in part by National Science Foundation grants IIS-1205249, IIS-1017134, CNS-0960061, EECS-0926052, the DARPA program on Autonomous Robotic Manipulation, the Office of Naval Research, the Okawa Foundation, and the Max-Planck-Society.

*http://www-clmc.usc.edu/Main/BharathSankaran

1 Introduction

For robots to be able to manipulate in unknown and unstructured environments the robot should be capable of operating under partial observability of the environment. Object occlusions and unmodeled environments are some of the factors that result in partial observability which in turn causes an uncertainty in the robot state estimate. A common scenario where this is encountered is *manipulation in clutter*. In the case that the robot needs to locate an object of interest and manipulate it, it needs to perform a series of decluttering actions to accurately detect the object of interest. To perform such a series of actions, the robot also needs to account for the dynamics of objects in the environment and how they react to contact. This is a non trivial problem since one needs to reason not only about robot-object interactions but also object-object interactions in the presence of contact. In the example scenario of *manipulation in clutter*, the state vector would have to account for the pose of the object of interest and the structure of the surrounding environment. The process model would have to account for all the aforementioned robot-object, object-object interactions. The complexity of the process model grows exponentially as the number of objects in the scene increases. This is commonly the case in unstructured environments. Hence it is not reasonable to attempt to model all object-object and robot-object interactions explicitly.

Also in some cases of human decision making we observe that we don't reason over all the possible agent-object and object-object interactions when manipulating in unstructured environments. For instance, imagine the case where you are looking for your keys on a table among clutter. When sifting through clutter we don't reason about all possible agent-object or object-object interactions. Since we have an accurate model of the object of interest, i.e the keys, we only reason about a limited set of cases. Such as the possibility of the keys being occluded by an object, etc. Under this setting we can formulate the problem as one where we construct a set of hypothesis about the possible poses of the object of interest given the current evidence in the scene and select actions based on our current set of hypothesis. This hypothesis set tends to represent the belief about the structure of the environment and the number of poses the object of interest can take. The uncertainty relating to the pose of the object of interest is directly dependent on the structure of the environment, i.e on the number other known or unknown objects in the environment. The agent's only stopping criterion is when the uncertainty regarding the pose of the object is *fully resolved*. The question to naturally pose is, is it possible to learn a search policy for such settings in real systems. Also what are the constraints that must be applied to the problem setting to make learning tractable. A crucial factor to note is, as the size of the environment grows, the size of this hypothesis set also grows.

2 Problem Formulation

Consider a robot that has access to a database of object models $\mathcal{O} = \{O_1, ..., O_n\}$ and a set of actions $\mathcal{A} = \{a_1, ..., a_K\}$. These actions could be movement primitives. Our task is to locate an object of interest $O_i \in \mathcal{O}$ in a cluttered environment. To accomplish this task, we need to execute a sequence of actions from $\mathcal{A}$ to manipulate the environment, to accurately detect $\mathcal{O}_i$. For this problem we denote our current state vector as $X_t \in \mathcal{X}$ which comprises of the pose of $\mathcal{O}_i$ represented by $\mathcal{P}_t \in \mathcal{P}$. $\mathcal{P}_t$ is dictated by an object model and the current structure of the environment $\mathcal{E}_t \in \mathcal{E}$. $\mathcal{E}_t$ is a voxelized representation where the occupancy of voxels are informed by the poses of all the other detected objects in the environment, whose shapes are dictated by object models or shape primitives. Let b denote the belief state, i.e. the distribution over the state space $\mathcal{X}$. Our objective is to learn a policy that will give us an action to execute given our current belief about the state. In essence we want to learn a policy $\pi : b(X_t) \rightarrow \mathcal{A}$, where $X_t = [\mathcal{P}_t; \mathcal{E}_t]$. To determine the optimal sequence of actions to achieve our task, we can formulate the problem as a POMDP, where our optimal policy would be given by

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(b(X_0))$$

where $b(\mathcal{X}_0)$ is our initial belief. The optimal policy, denoted by π^* yields the highest expected reward value for each belief state, which is represented by an optimal value function V^*. This value function can be calculated as

$$V^*(b(X_t)) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(b(X_t), a) + \gamma \sum_{\mathrm{Z_t} \in \mathcal{Z}} O(Z_t | b(X_t), a) V^*(\tau(b(X_t), a, Z_t)) \right]$$

Here γ is a discount factor and our reward is defined as:

$$\mathcal{R}(b(X_t), a) = \begin{cases} 1 & \text{if } a = a_{\text{ter}} \\ 0 & \text{otherwise} \end{cases}$$

An action $a = a_{\text{ter}}$ if the object of interest is successfully located. In this formulation we also assume access to an observation model $O(Z_t | b(X_t), a)$ and a belief space process model $\tau(b(X_t), a, Z_t)$, i.e we can accurately predict the outcome of an action. The belief model process model in this formulation inherently assumes one of two criteria. Either we can model the dynamics of interactions between various rigid bodies in the environment or we can model the evolution of

the hypothesis set as an outcome of actions executed. As mentioned earlier in Section 1, both of these tasks are non trivial. Given the context of our problem it is not easy to model object-object and robot-object interactions *or* model the change in the state uncertainty as an outcome of physical interaction. A possible argument to model either of these phenomena would be to learn from demonstrations or synthetic data. Even if we were to learn these distributions from demonstrations or synthetic data, the number of samples required to reasonably approximate the state space would be exponential in the number of objects in the environment. A similar argument can be made for the observation model. Also, the belief function $b(X_t)$ is hard to estimate given a large state space, as it needs to account for the object pose $\mathcal{P}$ and the entire structure of the environment $\mathcal{E}$. Hence, we constrain this general formulation.

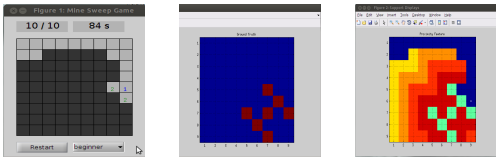
We note that we can in principle filter the belief using Bayesian filtering to account for the entire history of observations and actions. In our case, the belief function $b()$ represents the distribution over the object poses and current structure of the environment. Note that the object poses are dependent on the structure of the environment hence modeling this uncertainty is not straightforward. Instead of parameterizing the distribution of the state vector X_t, we adopt a non parameterized approach where we use a discrete set of hypotheses $\mathcal{H} = \{H_1, ..., H_m\}$ that can be constructed using the model of our object of interest $\mathcal{O}_i$ and the current state of the environment $\mathcal{E}_t$. The state of the environment at time t is estimated from observation Z_t given by a visual sensor. Given our current observation Z_t, we specify the belief $b(X_t)$ as the current hypotheses object poses with respect to the visible environment, given by the set $\mathcal{H}_t = b(X_t)$. This hypothesis set is constructed using tools from vision that take the object model $\mathcal{O}_i$ and observation Z_t and return $\mathcal{H}_t = \phi(Z_t, \mathcal{O}_i)$. The objective of the problem is to manipulate the environment till we have reduced the cardinality of our current hypothesis set to 1, $\|\mathcal{H}_t\| = 1$ so that we can successfully execute a model based manipulation action. We define this action as a terminal action $a_{ter} \in \mathcal{A}$ with reward 1. In an effort to make learning and inference in this setting tractable, we approximate quantities that can easily observed and modeled. Instead of trying to learn the dynamics of interactions in the environment, we try to directly learn a mapping between the belief state $b(X_t)$ and actions $\mathcal{A}$. This mapping is learned with discriminative classifiers that return an action given the current belief state. To ensure that the state space of the problem does not grow exponentially with the number of objects in the scene, we make the classifiers agnostic to the complete state of the environment and instead have them classify actions based on features computed on the current hypothesis set $\mathcal{H}_t$. We assume that we can construct the hypothesis set for any object model $\mathcal{O}$ under any observation in $\mathcal{Z}$, i.e $\mathcal{H} = \phi(Z_t, \mathcal{O}_i)$. Hence our policy learning problem is reduced to

$$\pi^* = \underset{a}{\text{argmax}}\ w^T f(b(X_t), a) \text{ where } b(X_t) = \mathcal{H}_t$$

Here different policies can be learned and compared by either altering the features or the number of classes, i.e actions.

3 Modified Minesweeper Simulation

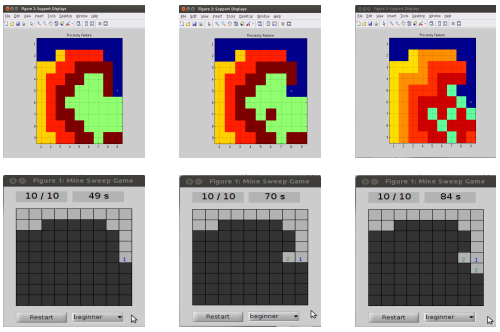
We emulate the problem of action selection under partial observability using a modified minesweeper scenario. In our modified minesweeper scenario, the mines are organized into a fixed size H-structure in the grid. The objective of the game is to accurately determine the pose of this hidden H-structure by opening a minimum number of non-mine cells. As in the classical minesweeper scenario opened cells may either be numbered or empty indicating the number of mines in the 8-connected neighbourhood *or* the opened cell might be a mine in which case the game terminates. The agent selects actions based on its current hypothesis set. This set is constructed based on the current observation, i.e opened cells and their values. The game is completed when the agent has narrowed down its set of hypothesis to one. The set of actions available to the agent is to open a cell from the 8-connected neighbourhood of the current open cell. The game play is initialized randomly.



(a) (b) (c)

Figure 1: Modified Minesweeper

A demonstration of this game play environment is show in Figure 1, where Figure 1a is the actual game play environment, Figure 1b is the ground truth location of the hidden H-structure and Figure 1c shows the features computed on the current hypothesis set. The feature we use is an inverse distance transform where cells close to the current set of hypothesis get a high score and cells far away from the hypothesis set get a low score. We then extract local templates from the features computed on the hypothesis set. These templates are 3x3 patches around the current expert location. The class corresponding to the feature is the location of the next action selected by the expert in the 8-connected neighbourhood. The evolution of the hypothesis set corresponding to the current game environment is demonstrated in Figure 2. We train the agent with demonstrations from an expert where the expert plays the game over a number of trials.



(d) (e) (f)

Figure 2: Hypothesis Set and Game Environment Updates

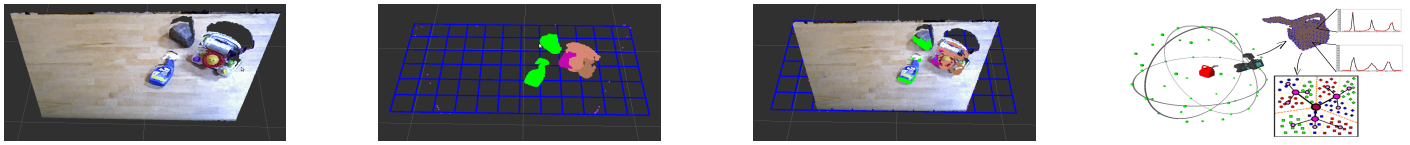
We compare different agents against a heuristic player (**HP**). The agents trained were a Multiclass (**MC**) 1 vs all SVM trained on the local templates with 8-connected neighbourhood as; a binary agent (**BE**) that classifies a local template from anywhere on the grid as actionable or not and a binary 8-connected (**B8**) agent that applies the binary agent to the 8-connected grid. We tested the various agents over 100 different trials with 10 random poses of the hidden H-structure and each of the 10 poses had 10 different initializations for the agent. The results are tabulated in Table 1. The results show the mean number of actions taken over the successful trials off the 10 trials. The number of failed attempts in these 10 trials are boldfaced. Failures result due to opening a mine or in the **B8** case failing to classify any neigbouring grid as actionable. The best result for each random pose are highlighted in green.

Agent	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10
MC	12.3	8.4;**2**	8.1	10.1	7.1;**1**	8.3	7.1;**1**	10.1	12.6	13.2;**2**
BE	13.8	11.3	13.2	16	10.6	15.6	20.8	11.6	12.1	14.5
B8	12.3;**3**	8.4;**2**	9.1;**3**	10.1;**2**	9;**4**	17.8;**4**	8.3;**3**	10.1;**2**	12.6;**5**	13.2;**3**
HP	25.6	9.3	8;**1**	6.4	7.8;**1**	13.8	8.3	7.1	28.5	21

Table 1: Results of Minesweeper Tests

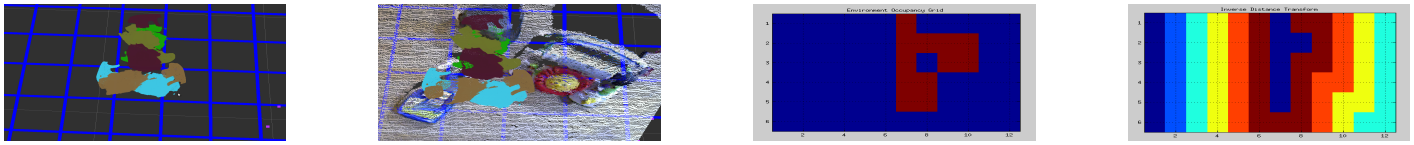
4 Transition to a Real Robot Environment

We apply the same policy learning framework to a real robot decluttering experiment, where the robot is tasked with locating an object of interest in a cluttered environment. Here the input observation $\mathcal{Z}_t$ is an RGBD pointcloud. The hypothesis set $\mathcal{H}_t$, of the object of interest is computed using the output of an object classifier [1], that returns an object class and pose hypothesis for every pointcloud cluster in the environment. These hypotheses are then projected on to a planar support surface (tabletop) to compute a hypothesis feature similar to the minesweeper scenario. The general pipeline is demonstrated in the figure below.



(a) Input Point Cloud (b) Pointcloud Clustering (c) Preprocessing Overlay (d) VP-Tree Classifier

Figure 3: Point cloud preprocessing



(a) Projected hypothesis (b) Hypothesis Overlay (c) Env Occupancy Grid (d) Inverse Dist Transform

Figure 4: Hypothesis Feature Computation

5 Conclusions and Future Work

We have demonstrated a policy learning approach for hypotheses based action selection. Our approach is trained in a supervised manner with expert demonstrations. The key features of our approach are we can accomplish complex tasks without reasoning about a process or observation model. Our approach also has the ability to scale to large environments and the learning complexity is agnostic to the size of the environment. Our proposed model simplification approach is only valid for the class of POMDP problems where states are strictly markovian in nature ex: [2, 3], i.e where the current observation encompases the history of all previous observations. In the future we are going to perform more tests on our robotic setup and apply this frame work to other policy learning tasks.

References

[1] N. Atanasov, B. Sankaran, J. Le Ny, G. Pappas, and K. Daniilidis, "Nonmyopic view planning for active object classification and pose estimation," 2014.

[2] S. Ross, G. Gordon, and J. A. D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the 14th International Conference on Artifical Intelligence and Statistics (AISTATS)*, April 2011.

[3] S. Ross and J. A. D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, May 2010.

[4] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. [Online]. Available: http://dl.acm.org/citation.cfm?id=645529.657801

[5] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. S. Srinivasa, "Object search by manipulation," in *ICRA*, 2013, pp. 4973–4980.

[6] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, Mar. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=944790.944813

Contingency and Correlation in Reversal Learning

Brad Pietras
Program In Neuroscience
University of Maryland
Baltimore, MD 15213 USA
brad.pietras@umaryland.edu

Thomas A. Stalnaker
National Institute on Drug Abuse
Intramural Research Program
251 Bayview Boulevard
Baltimore, MD 221224 USA
thomas.stalnaker@nih.gov

Tzu-Lan Yu
Program In Neuroscience
University of Maryland
Baltimore, MD 15213 USA
r90227001@ntu.edu.tw

Geoffery Schoenbaum
National Institute on Drug Abuse
Intramural Research Program
251 Bayview Boulevard
Baltimore, MD 221224 USA
geoffrey.schoenbaum@nih.gov

Peter Dayan
Gatsby Computational Neuroscience Unit
University College London
London, UK
dayan@gatsby.ucl.ac.uk

Abstract

Reversal learning is one of the most venerable paradigms for studying the acquisition, extinction, and reacquisition of knowledge in humans and other animals. It has been of particular value in asking questions about the roles played by prefrontal structures such as the orbitofrontal cortex (OFC). Indeed, evidence from rats and monkeys suggests that these areas are involved in various forms of context-sensitive inference about the contingencies linking cues and actions over time to the value and identity of predicted outcomes. In order to explore these roles in depth, we fit data from a substantial behavioural neuroscience study in rodents who experienced blocks of free- and forced-choice instrumental learning trials with identity or value reversals at each block transition. We constructed two classes of models, fit their parameters using a random effects treatment, tested their generative competence, and selected between them based on a complexity-sensitive integrated Bayesian Information Criteria score. One class of 'return'-based models was based on elaborations of a standard Q-learning algorithm, including parameters such as different learning rates or combination rules for forced- and fixed-choice trials, behavioural lapses, and eligibility traces. The other novel class of 'income'-based models exploited the weak notion of contingency over time advocated by Walton et al (2010) in their analysis of the choices of monkeys with OFC lesions. We show that income-based and return-based models are both able to predict the behaviour well, and examine their performance and implications for reinforcement learning. The outcome of this study sets the stage for the next phase of the research that will attempt to correlate the values of the parameters to neural recordings taken in the rats while performing the task.

Keywords: reversal learning, orbitofrontal cortex, income, return, Q-learning

Acknowledgements

We are grateful to Robert Wilson for helpful comments. Funding was from the Gatsby Charitable Foundation (PD) and the U.S. National Institute on Drug Abuse Intramural Research Program.

1 Introduction

It has been known for over four decades that the OFC makes an important contribution to flexibility in decision-making [2], particularly when changes in stimulus-reward contingencies require the subject to change behavior in order to maintain optimal performance. Contingency emerges from the coupling between choices and rewards; how this relationships are measured and accumulated over trials is as yet not completely clear. We consider two studies that examined influences of both choice and reward patterns. The first involved macaque monkeys performing a three-armed bandit task with stimulus-reinforcement contingency reversals before and after OFC lesions [5]. OFC lesions did, as expected, impair the subjects' performance after the contingency-reward reversal. However, careful examination of their choices revealed just a subtle impairment of the subject's ability to assign a reward to the single correct choice. Subjects instead associated these rewards with the series of choices made in the recent past. This would allow the subject to make correct choices for times when the same choice leads to a reward, but performance after a contingency reversal would still be greatly impaired. It can also be seen as being reminiscent of what is known as an income-, rather than a return-based, method for determining preference [3]. In the second study, rats performed an odor-guided instrumental learning task in which the magnitude and identity of the rewards were subject to reversals. Behaviour was recorded together with neural activities in the OFC [4] and ventral striatum (VS). Here, we model the rats' behavior with both return- and income-based learning rules inspired by the monkey study.

2 Experimental Methods and Data

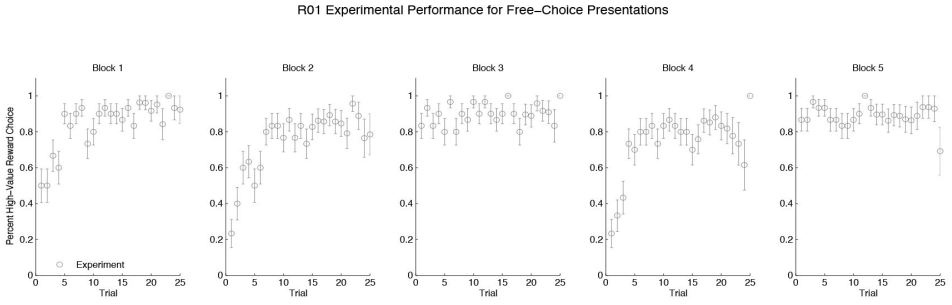


Figure 1: Experimental performance of rat R01 in odour-directed task for free-choice presentations. Trials averaged across 30 sessions. Cue-response contingency changes occur at the onset of each block in a sequence of value-identity-value-identity reversals. These trials are interspersed amongst forced trials which also provide evidence about the consequences of choices.

We modelled behavioral data from rats ($n = 13$) trained to perform an odor-guided instrumental learning task with three odor cues [4]. Trials commenced with a head-poke into a central well followed by receipt of an odour indicating the possible rewards from left and right wells. Two odour cues were associated with "forced" trials, in which the subjects could only get reward by going either left or right appropriately – each well was associated with either a small (one drop) or large (three drops) of reward for a block. Randomly mixed (7/20) with the forced trials were "free" trials indicated by a third odor cue, in which both wells were 'armed' with the same reward available on the forced trials. The reward identity was either chocolate or vanilla flavored milk, which the subjects liked equally. The cue-reward contingency was fixed for each block but then changed at each block transition in a sequence of *value-identity-value-identity* reversals across the next four consecutive blocks of trials. Block transitions were partly occasioned by satisfactory performance of the subjects; the first block lasted 43 ± 16 trials; subsequent blocks lasted 63 ± 10.7. Behavior and single unit neural activity in non-lesion and unilaterally lesioned OFC and VS were recorded simultaneously.

Figure 1 shows rat R01's average performance over 30 sessions of the odour-directed task across five sequential blocks of free trials. As can be seen in the figure, the rat begins the free trial presentations in block 1 with a 50% probability of choosing the higher valued reward, but by the fifteenth trial is on average selecting this option with about 90% probability. After the first reversal of reward value at the onset of block 2, the performance of the rat drops to well below 50%, but again, by the fifteenth trial, has recovered to selecting the larger reward with about 80% probability. The rat's performance continues to improve through the identity reversal in block 3 but repeats the pattern seen in blocks 2 and 3 after the second value reversal in block 4 and identity reversal in block 5.

3 Learning Rules

3.1 Introduction

We consider two classes of learning rule. The first involves variants of regular, contingent Q-learning, in which the value of a choice is only dependent on the reward received for that choice. The second class, correlational Q-learning, measures a temporally-delocalized correlation between the stream of rewards earned by the subject and the stream of choices made, without monitoring the detailed contingency between the two. We lack the space to discuss the many variants of these learning rules that we also considered.

3.2 Contingent Q-learning: Q2

To fix the notation, we consider a return-based Q^R-learning rule with two parameters: $\theta^R = (T^R, \alpha^R_Q)$, where T^R is the temperature associated with a softmax choice and α^R_Q is the learning rate for Q^R-values. The probability of choosing action a_t on trial t, and the update to Q^R_t based on the prediction error ΔQ^R_t are given by

$$p(a_t|\theta; Q^R_t) = e^{\frac{Q^R_t(a_t)}{T^R}} / \sum\nolimits_b e^{\frac{Q^R_t(b)}{T^R}} \qquad Q^R_{t+1}(b) = Q^R_t(b) + \alpha^R_Q \Delta Q^R_t(b) \qquad \Delta Q^R_t(b) = r_t - Q^R_t(a_t), \forall b \tag{1}$$

3.3 Correlational Q-learning: QB5

We designed a novel, income-based, learning rule based on the correlation between low-pass filtered rewards and choices that captures the impaired causal relationship between rewards and choices following OFC lesions in macaques [5]. This involves three additional parameters (now labelled with the superscript I) beyond contingent Q^R-learning as follows: $\theta^I = (T^I, \alpha^I_Q, \alpha^I_C, \chi^I_C, \lambda^I_Q)$ where T^I is the temperature associated with a softmax choice, α^I_Q and α^I_C are the learning rates for Q^I-values and the choice kernel respectively, χ^I_c is the weight of the choice kernel and λ^I_Q is an eligibility trace parameter. The critical difference from contingent Q^R-learning is the prediction error $\Delta Q^I_t(b)$:

$$\Delta Q^I_t(b) = \rho_t \frac{C_t(b)}{\sum_d C_t(d)} - Q^I_t(b) \qquad \text{with} \qquad \rho_{t+1} = \rho_t + \alpha^I_C(r_t - \rho_t) \tag{2}$$

where the probability of choosing action a_t on trial t, the updated Q^I, the eligibility trace $\mathbf{e}^I_t$ and the choice kernels C_t are updated according to

$$p(a_t|\theta^I; Q^I_t; C_t) = e^{\frac{Q^I_t(a_t)+\chi^I_c C_t(a_t)}{T^I}} / \sum\nolimits_b e^{\frac{Q^I_t(b)+\chi^I_c C_t(b)}{T^I}} \qquad Q^I_{t+1}(b) = Q^I_t(b) + \alpha^I_Q \Delta Q^I_t(b)(1-\lambda^I_Q) e^I_t(b) \tag{3}$$

$$e^I_{t+1}(b) = \lambda^I_Q \left(e^I_t(b) + \delta_{a_t b}\right) \qquad \text{and} \qquad C_{t+1}(b) = \begin{cases} (1-\alpha^I_C)C_t(b) + \alpha^I_C(1 - C_t(b)) & \text{if } a_t = b \\ (1-\alpha^I_C)C_t(b) & \text{otherwise} \end{cases} \tag{4}$$

4 Model Results and Evaluation

By including or excluding parameters (for instance, eliminating the eligibility trace), both classes of models can be seen as a lattice of possibilities. We fit the resulting set of models to the data using the approximate type II maximum likelihood scheme in [1]. That is, we treat the subjects as 'random effects', i.e., with a hierarchical generative model involving top-level independent Gaussian distributions associated with each parameter. We fit the means and standard deviations of these distributions using the expectation-maximization algorithm in conjunction with a Laplace approximation for the posterior distributions for each subject. We transformed the Gaussian random variables via non-linear functions $\exp(\cdot)$ (for the temperature) and $\sigma(\cdot) = \epsilon + (1-2\epsilon)/(1+\exp(-\cdot))$ (for the other parameters, where ϵ = 0.010) to keep them in the appropriate ranges. We compare models using the integrated Bayesian Information Criteria (iBIC) [1] score, which combines the likelihood of the choices under the full generative model with a complexity penalty.

There are various ways to incorporate the forced trials into this analysis. Unsurprisingly, the subjects perform these trials very well; however, although they do provide information about the reward availability on each side, this information might not be associated by the subjects with the odor that signals a free choice. We are currently exploring the best way to incorporate them; all the results here exclude them from consideration entirely.

Figures 2 and 3 illustrate the workings of the two representative models above: return-based Q2 and income-based QB5.

We generated simulated performance data by sampling values of model parameters for each session via the non-linear transforms from the top-level Gaussian distributions, using these to compute choice probabilities on a trial-by-trial

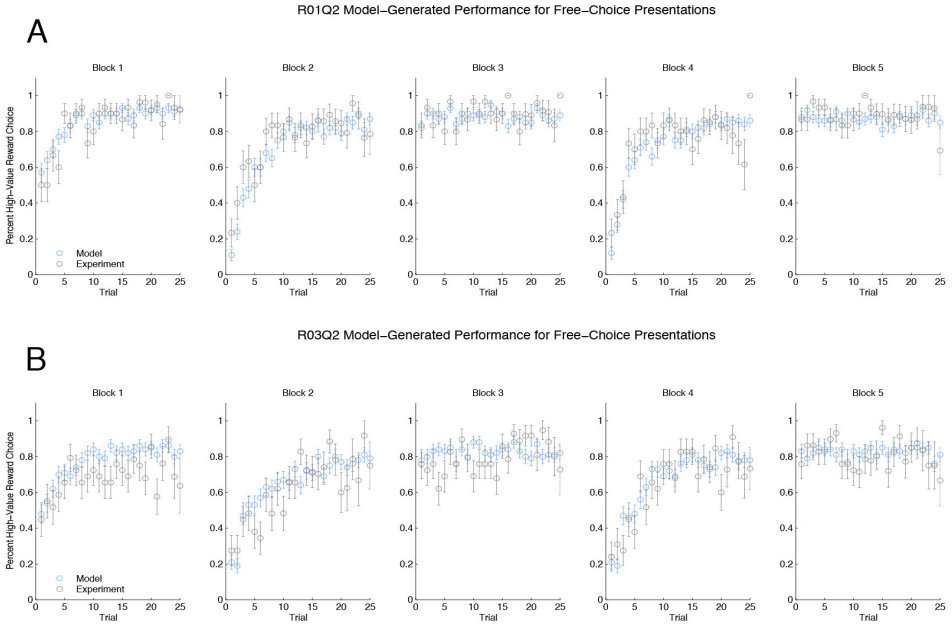


Figure 2: Q2 results for rats R01 (A) and R03 (B) showing average performance across 100 sessions with parameters drawn from hyperparameter means $\pm$ standard deviation in non-linear coordinates. R01: $T^R = \exp(0.005 \pm 0.031)$,$\alpha^R = \sigma(-0.293 \pm 1.000)$; R03: $T^R = \exp(0.241 \pm 0.042)$, $\alpha^R = \sigma(-0.530 \pm 1.000)$

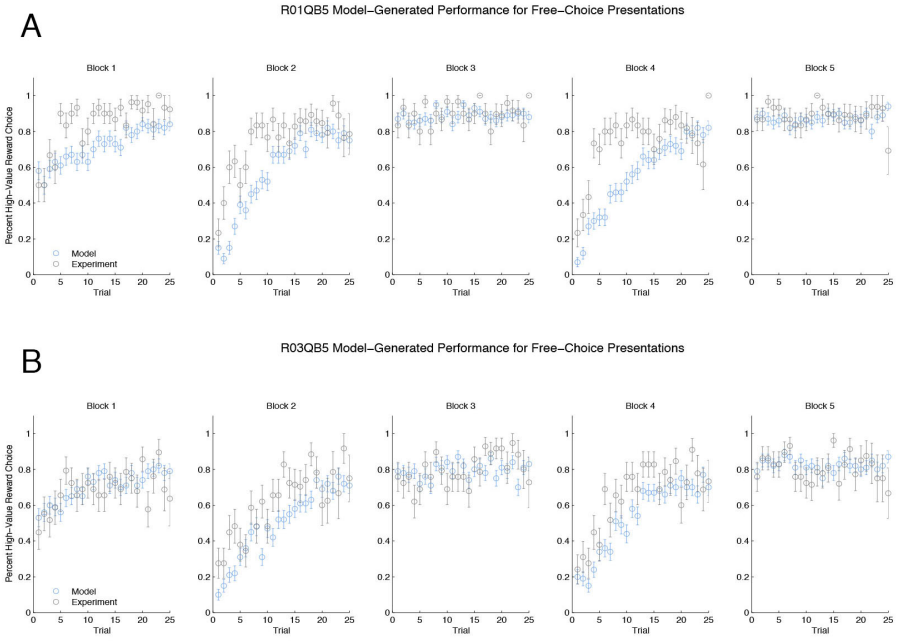


Figure 3: QB5 results for rats R01 (A) and R03 (B) showing average performance across 100 sessions with parameters drawn from hyperparameter means $\pm$ standard deviation in non-linear coordinates. R01: $T^I = \exp(0.215 \pm 0.002)$ $\alpha^I_Q = \sigma(0.197 \pm 0.072)$, $\alpha^I_C = \sigma(2.400 \pm 0.147)$, $\chi^I_C = \sigma(1.341 \pm 0,261)$, $\lambda^I_Q = \sigma(-0.982 \pm 1.300)$, R03: $T^I = \exp(0.319 \pm 0.005)$ $\alpha^I_Q = \sigma(-0.193 \pm 0.650)$, $\alpha^I_C = \sigma(1.781 \pm 0.248)$, $\chi^I_C = \sigma(0.658 \pm 0.194)$,$\lambda^I_Q = \sigma(-1.420 \pm 0.248)$

basis, and then picking choices from these probabilities. Both models were able to reproduce the experimental results well. Figure 4 shows iBIC scores for the two models for each of two rats. For subject R01, the Q2 model (normalized iBIC score = 0.8752) had a slight performance advantage over QB5 (normalized iBIC score = 0.8793) but for subject

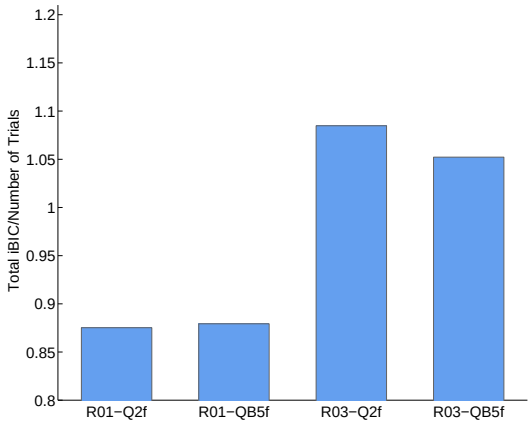


Figure 4: Integrated BIC scores for rats R01 and R03. Q2f and QB5f use only free trials when computing the parameter means and variances with expectation maximization and for generating simulated performance.

R03, who demonstrated a much slower recovery from reversals than rat R01, QB5 had better performance (normalized iBIC score = 1.0522) compared to Q2 (normalized iBIC score = 1.0848). Moreover, Q2 recovered slightly faster from reversals than QB5, occasionally outperforming the experimental subject. Interestingly, rat R03 recovered from the second reversal slightly faster than from the first. We are currently studying individual sessions to see if, contrary to the apparent sloth of learning overall, there is evidence for the sort of near-instantaneous state-switching suggested by Wilson et al [6].

5 Conclusion

Reversal learning has been a touchstone for understanding flexible behaviour and is a significant target for work in OFC. Inspired by findings from macaques with OFC lesions [5], we introduced a new class of Q-learning rules we call correlational Q-learning that takes into account the possibility for contingency mis-assignment between a series of outcomes and the choices that lead to those outcomes. We compared this class with standard, contingent, Q learning algorithms on the behavior of rats in an odor-guided instrumental learning task with value and identity reversals. We chose 2 rats out of our population of 13, one each fit better by the two rules; performance overall was quite comparable. The next steps for behavioural modeling are to study the integration of forced trials with the free ones, assessing the possibility of adaptive learning rates or rapid context switches following reversals [6] and systematically to test the benefits of all the parameters. We will thus obtain a rich set of potential correlates with which to analyze the simultaneously-collected electrophysiological data.

References

[1] Q.J.M. Huys, N. Eshel, E. O'Nions, L. Sheridan, P. Dayan, and J.P Roiser. Bonsai trees in your head: How the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS Computational Biology*, 8,3, March 2012.

[2] B. Jones and M. Mishkin. Limbic lesions and the problem of stimulus-reinforcement associations. *Experimental Neurology*, 36:362–377, 1972.

[3] B. Lau and P.W. Glimcher. Dynamic response-by-response models of matching behavior in rhesus monkeys. *Journal of the Experimental Analysis of Behavior*, 84:555–579, November 2005.

[4] T.A. Stalnaker, N.K. Cooch, M.A. McDannald, T.L. Liu, H. Weid, and G. Schoenbaum. Orbitofrontal neurons infer the value and identity of predicted outcomes. *Nature Communications*, 5:3926, 2014.

[5] M.E. Walton, T.E.J. Behrens, M.J. Buckley, and P.H. Rudebeck. Separable learning systems in the macaque brain and the role of the orbitofrontal cortex in contingent learning. *Neuron*, 65:927–939, March 2010.

[6] R.C. Wilson, Y.K. Takahashi, G. Schoenbaum, and Y. Niv. Orbitofrontal cortex as a cognitive map of task space. *Neuron*, 81:267–279, January 2014.

A Drift Diffusion Model of Proactive and Reactive Control in a Context-Dependent Two-Alternative Forced Choice Task

Olga Lositsky
Princeton Neuroscience Institute
Princeton University
lositsky@princeton.edu

Robert C. Wilson
Department of Psychology
University of Arizona
bob@email.arizona.edu

Michael Shvartsman
Princeton Neuroscience Institute
Princeton University
ms44@princeton.edu

Jonathan D. Cohen
Princeton Neuroscience Institute
Princeton University
jdc.princeton.edu

Abstract

Most of our everyday decisions rely crucially on context: foraging for food in the fridge may be appropriate at home, but not at someone else's house. Yet the mechanism by which context modulates how we respond to stimuli remains a topic of intense investigation. In order to isolate such decisions experimentally, investigators have employed simple context-based decision-making tasks like the AX-Continuous Performance Test (AX-CPT). In this task, the correct response to a probe stimulus depends on a cue stimulus that appeared several seconds earlier. It has been proposed (Braver, 2007) that humans might employ two strategies to perform this task: one in which rule information is *proactively* maintained in working memory, and another one in which rule information is retrieved *reactively* at the time of probe onset. While this framework has inspired considerable investigation, it has not yet been committed to a formal model. Such a model would be valuable for testing quantitative predictions about the influence of proactive and reactive strategies on choice and reaction time behavior. To this end, we have built a drift diffusion model of behavior on the AX-CPT, in which evidence accumulation about a stimulus is modulated by context. We implemented proactive and reactive strategies as two distinct models: in the proactive variant, perception of the probe is modulated by the remembered cue; in the reactive variant, retrieval of the cue from memory is modulated by the perceived probe. Fitting these models to data shows that, counter-intuitively, behavior taken as a signature of reactive control is better fit by the proactive variant of the model, while proactive profiles of behavior are better fit by the reactive variant. We offer possible interpretations of this result, and use simulations to suggest experimental manipulations for which the two models make divergent predictions.

Keywords: cognitive control; context-dependent decision-making; proactive and reactive control; dual-mechanisms of control; drift diffusion model

Acknowledgements: This project was made possible through the support of a grant from the John Templeton Foundation. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the John Templeton Foundation.

1 Introduction

Most of our daily decisions depend crucially on context. While our normal response to seeing a new email from a good friend might be to open it, this response may be suppressed if our goal is to work. However, if an email alert were to arrive about an approaching winter storm, we should be able to respond to it despite its former irrelevance. This ability to modulate our behavior based on context has been termed "cognitive control." The traditional account of these context effects is a *proactive* one, in which context is represented by patterns of ongoing neural activity (possibly sustained by the frontoparietal network) and modulates the response to incoming stimuli (Miller & Cohen, 2001). Recently, the Dual Mechanisms of Control (DMC) account has proposed that, since active task preparation is costly, people may sometimes use a *reactive* strategy. Instead of actively maintaining the context, it could be stored passively (perhaps in a pattern of synaptic weights) and reactivated when the relevant stimulus is presented (Braver, 2007). Under the DMC framework, people can switch between these two strategies based on their intrinsic motivation, the constraints of their cognitive architecture, and the particular demands of the task (Braver, 2007).

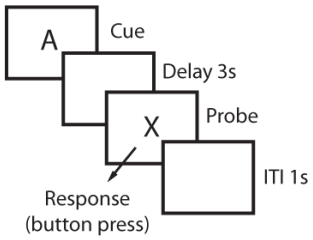
While the DMC framework has inspired a large body of experimental work in humans, we provide the first computational instantiation of it, and apply it to the AX-Continuous Performance Test (AX-CPT), a classic task from the cognitive control literature. In the AX-CPT, a contextual cue (A or B) is presented briefly on each trial, followed by a probe stimulus (X or Y) after a short delay. Participants must respond to the probe stimulus differently depending on whether it was preceded by (i.e. in the context of) an A or a B.

We model proactive control as identifying the probe (X or Y) in the context of the cue (A or B), and reactive control as recalling the cue (A or B) in the context of the probe (X or Y). We instantiate the probe identification or cue recall processes as drift-diffusion models, which allow us to predict the pattern of choices and reaction times for the different trial types in the task. Using these models, we show that choice behavior taken as a signature of proactive or reactive control can be produced by both instantiations of our model, though they make different predictions for reaction time distributions. Moreover, we find – counter-intuitively – that proactive behavior is better fit by the model we termed "reactive", while reactive behavior is better fit by the model we termed "proactive", and offer possible interpretations. Our simulations suggest that manipulating uncertainty about the cue and probe separately could bring subjects into a parameter regime where the two models make the most distinct predictions.

2 Methods

2.1 The AX-CPT Task

On each trial of the task, a cue (A or B) signals how to respond to the upcoming probe (X or Y), which appears a few seconds later. In the variant we use, the subject must press the left button on AX and BY trials, and the right button on AY and BX trials (Fig. 1). Manipulating the frequencies of cue and probe stimuli can be used to investigate subjects' control strategy. Since AX trials are the most common, a subject proactively preparing responses will be biased to incorrectly respond Left on AY trials, since X is more likely than Y. On the other hand, a subject reacting to the probe will be biased to incorrectly respond Left on BX trials, since A precedes X more often than B. The relative proportion of AY and BX errors and reaction times in such designs are therefore taken to reflect an index of proactive or reactive control (Braver, 2009).



Trial Type	Freq.	Response
AX	50%	Left
AY	20%	Right
BX	20%	Right
BY	10%	Left

Figure 1 Trial timeline, trial frequencies and response rules.

2.2 Behavioral Experiments to Induce Proactive and Reactive Strategies

To encourage subjects to use both types of control, we used a number of within-subject manipulations. In the proactive manipulation, subjects were rewarded for correctly responding faster than 500ms, which is difficult without proactive task preparation. In the reactive manipulation, we introduced a distractor task between the cue and probe, making it more difficult to actively maintain the context during the delay period, such that subjects might have to retrieve their context representation at the time of the probe.

We ran 11 small experiments, varying trial frequencies (while maintaining AY and BX frequencies equal), the duration of the delay period, and the difficulty of the second task. While these variations are interesting and will be reported separately, here we aggregate the results from all experiments (120 subjects) in order to draw general inferences about the proactive-bias and reactive-bias conditions.

2.3 Drift Diffusion Model of Context-Dependent Decision-Making on the AX-CPT

At its heart, the AX-CPT is a two-alternative forced choice (2AFC) task: participants make a decision between left and right on the basis of a context cue (A or B) and a probe stimulus (X or Y). Such 2AFC tasks have been modeled successfully using drift diffusion models (DDMs), in which a decision is made by gradually accumulating evidence for either left or right response and responding when the evidence crosses some decision threshold, z (see Bogacz, 2006, for a detailed review). Unlike classic 2-AFC tasks, however, in the AX-CPT, subjects must integrate memory of the cue they saw seconds earlier with incoming perceptual evidence about the probe, and the dynamics of this integration should vary by strategy. We implemented the proactive strategy as a drift diffusion process reflecting perceptual evidence accumulation about the probe, the parameters for which are set by the noisily encoded cue in memory. We modeled the reactive strategy as a drift diffusion process reflecting retrieval of the cue from memory, the parameters of which are set by noisy perception of the probe. This distinction reflects the DMC conception of proactive control as task configuration triggered by the cue and reactive control as task configuration triggered by the probe.

2.3.1 Drift Diffusion-on-Probe (DDP) model of "proactive" strategy

In the proactive variant, which we will term "Drift Diffusion-on-Probe" (DDP), the true cue, C, is encoded in memory as $\tilde{C}$, with some bit-flip noise, ε_C, which reflects the probability that the cue is encoded incorrectly. Given its noisy cue representation, the model computes the posterior probability distribution over the rule, R (i.e. R = "left on X, right on Y" or "right on X, left on Y") using Bayesian inference: $P(R|\tilde{C}) \propto P(\tilde{C}|R)P(R)$, with the prior set based on trial frequencies, and the likelihood computed using the true value of ε_C.

The model then samples the rule from $p(R|\tilde{C})$, and uses this sampled rule to set up the parameters of the DDM on the probe. The sampled rule determines the direction of the drift, the threshold, z, and starting point, x_0, for drift diffusion on the probe. The model therefore has two starting point and threshold parameters (one for each cue: $\tilde{z}^A, \tilde{z}^B, {x_0}^A, {x_0}^B$), a drift rate parameter for the probe ($\tilde{a}$), cue noise (ε_C) and a response time offset due to motor planning and execution (T0) – seven free parameters in total.

These assumptions, in combination with standard results from the DDM literature (Busemeyer & Townsend, 1993; Navarro & Fuss, 2009), allow us to compute reaction time distributions for correct and error trials in each of the four trial types (AX, AY, BX, BY). This, in turn, allows us to fit the model to the experimental data.

To develop an intuition about how the model behaves, consider that the probability of using Rule B given that $\tilde{C}$=B increases with the frequency of B cues and decreases with encoding noise, ε_C. As ε_C increases, the model will be more likely to confuse B for A cues, given the higher frequency of A. This will cause it to respond incorrectly on BX, and produce the typically "reactive" mistake. In other words, the higher the cue noise, the more reactively our "proactive" model behaves.

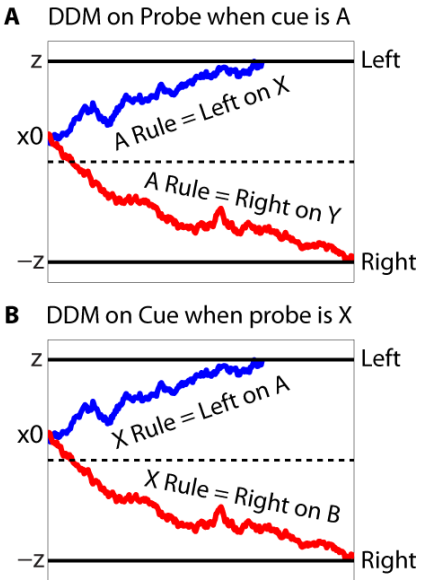


Figure 2 Proactive DDM on Probe (A) and Reactive DDM on Cue (B)

2.3.2 Drift Diffusion-on-Cue (DDC) model of "reactive" strategy

For the reactive variant, which we term "Drift Diffusion-on-Cue" (DDC), we reversed this process to reflect the fact that the reactivation of task representations is triggered and modulated by the probe. In this variant, the probe is encoded with some bit-flip noise ε_P, and must be decoded in order to set the rule (e.g. Rule = "Right on A, Left on B".) As the probe encoding noise increases, the model will be more likely to confuse Y's for X's, given the higher frequency of X's, causing it to respond incorrectly on AY trials. By a similar intuition as above, our "reactive" model will produce typically "proactive" mistakes as probe noise increases. The sampled rule also sets the starting point for drift diffusion on the cue. Given the trial frequencies (Fig. 1), a rational subject might set the X starting point closer to the Left, but the Y starting point closer to the Right threshold.

Note that both variants of the model implement noise in the cue and probe representations, but they do so in different ways. In the DDC model, noise in the *cue* representation would be reflected in a low drift rate, causing impaired BX performance. In the DDP model, a low drift rate would reflect noise in *probe* perception, impairing AY performance, while the cue encoding noise ε_C facilitates BX errors.

2.4 Fitting our Model to Data

We used maximum likelihood estimation to fit the DDP and DDC models separately for each subject in each experimental condition. The Bayesian Information Criterion can show us which model fits the data best in which condition, but does not allow us to assess the quality of that fit. We therefore computed a "pseudo R^2" statistic (Fernandes, 2014), which measures how well our models fit the data relative to a null model – with minimal assumptions – and a "saturated model" that overfits the data. Following Fernandes (2014), we defined the pseudo R^2 as:

$$pR^2 \stackrel{\text{def}}{=} 1 - \frac{L_S - L_T}{L_S - L_H}$$

where $L_S$is the log likelihood of the saturated model, L_T is the log likelihood of the theoretically-informed model, and $L_H$is that of the homogeneous or null model. These benchmark models should reflect the most and least constrained forms of our models. The homogeneous model assumes no variation by trial type and assumes all trials are drawn from a mixture of two arbitrary DDMs. The saturated model assumes no shared properties between trial types and assumes each trial type is drawn from a separate mixture of two arbitrary DDMs. (We make the mixture-of-DDMs assumption because our models produce a mixture of two DDMs by occasionally sampling from the wrong rule due to encoding noise.) The resulting saturated model has 32 parameters, and thus many more degrees of freedom than our 7-parameter models.

A pseudo R^2 value of 0 means our theoretical model does no better than a model that does not distinguish between trial types, and a value of 1 means that our model performs as well as a model that fits each trial type separately.

2.5 Landscaping Analysis: Finding where the models make distinct predictions

To infer in which part of the parameter space the DDP and DDC models make the most distinct predictions, we performed a "landscaping analysis" (Navarro, Pitt & Myung, 2004): we simulated behavior with each model, sampling uniformly from the parameter space, and fit this behavior with both models (repeating with both models as the simulating model). This analysis lets us identify parts of the parameter space where one model cannot fit the other's generated data, and therefore the models make distinct predictions.

3 Results

3.1 Behavioral Results

As Figure 3 shows, the proactive manipulations generally succeeded at eliciting more AY than BX errors ($t(119)= 3.35$, $p=0.001$), while the reactive manipulations were more variable at eliciting more BX than AY errors ($t(119)=1.89$, $p=0.06$). While mean response times varied by trial type (AY errors were faster than AY corrects, while AX errors were slower than AX corrects), they did not vary by condition, possibly because the incentive to respond faster in the proactive condition reduced the overall variability in decision times.

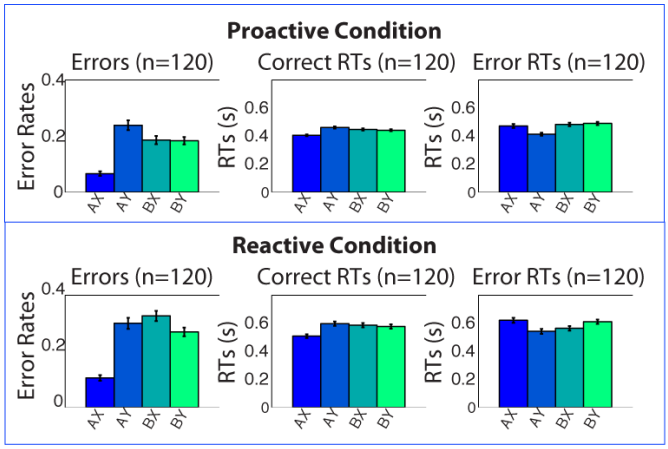


Figure 3 Mean behavior in the two experimental conditions.

3.2 Model Goodness of Fit by Experimental Condition and Model Type

The mean pseudo R^2 values ranged between 0.68 for DDP (SD=0.26) and 0.69 for DDC (SD=0.28) (Fig. 4A). Since the saturated model had four times as many parameters, our models received a consistently lower BIC than the saturated model (all $p < 10^{-10}$), suggesting that our models are a parsimonious description of the data. We discuss more detailed results next.

3.2.1 DDM on Cue Model Fits Proactive Behavior Better; DDM on Probe Model Fits Reactive Behavior Better

Considering only the proactive condition, the DDC model fit behavior significantly better than DDP ($t(119)=4.1$, $p < 10^{-4}$), while there was no difference between models for the reactive condition (t(119)=0.07, $p=0.94$). However, since there were large individual differences in how well the reactive manipulation elicited the desired behavior, we also split subjects into two groups based on their "proactive error index"

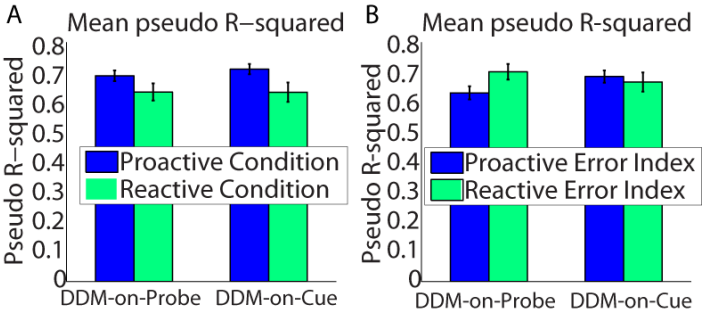


Figure 4 Quality of model fits by condition and proactive index

(AY errors – BX errors.) This revealed that proactive behavior was significantly better fit by the DDC than the DDP model ($t(110)$=10.00, $p < 10^{-16}$) while reactive behavior was better fit by the DDP than the DDC model ($t(106)$= 3.64, $p < 10^{-3}$, see Fig. 4B). This pattern was the opposite of what we had predicted, but can be explained by the fact that both models can in fact produce both types of behavior. Given that the DDC model represents cue noise through a low drift rate, it would need broad reaction time distributions in order to produce BX errors. This was not the case in our data, where subjects were encouraged to react quickly. Therefore, the DDP model was able to better capture BX errors through its bit-flip cue noise parameter (which does not affect reaction times.) Similarly, the DDC model's probe noise parameter could better account for proactive AY errors. This effect is further exacerbated by the lack of reaction time differences across conditions in our data (AY trials were slower than BX in both conditions). In such cases, the encoding noise parameter is better suited for producing error changes without large corresponding RT differences. We are currently running experiments without an RT deadline, to see if this effects holds when we induce more natural variability in reaction time distributions. Realistically, subjects may not adopt purely cue-based or probe-based strategies, and evidence accumulation might happen on both stimuli simulataneously at the time of response. We are currently building models that capture this interaction.

3.2.2 Behavior in proactive condition was better fit by all models than behavior in reactive condition

Both models received higher pseudo R^2 scores for the proactive condition than the reactive condition ($t(119)$=2.09, p=0.04 for DDP and t(119)= 2.73, p=0.007 for DDC, see Fig. 4A). Given that we did not reward speed or accuracy in the reactive condition, participants were likely more unstable in their strategy across trials. Such behavior is better fit by the saturated model, since it fits a weighted sum of two arbitrary DDMs. However, splitting behavior by the proactive error index did not reveal lower pseudo R^2 scores for reactive behavior ($t(216)$=0.46, p=0.64 for DDC), and the DDP model even trended towards fitting reactive behavior better than proactive behavior ($t(216)$=1.96, p=0.05), suggesting that the poorer fit was not related to higher BX errors, but rather specific to our reactive manipulation. Thus, while the reactive *manipulation* may have promoted more noisy behavior, our models can fit reactive profiles of behavior as well as proactive ones.

3.2.3 Landscaping analysis points to more sensitive experimental manipulations for model identification

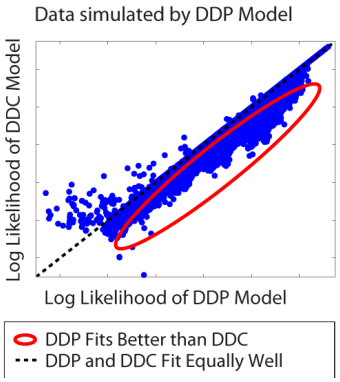


Figure 5 Landscaping Analysis

Besides removing the cap on reaction times, our simulations suggest experimental manipulations that can help to test the predictions of our two models more precisely. Comparing the log likelihoods of the two models (on data generated by one of them) for various parameter combinations revealed that they make the most distinct predictions when both encoding noise and drift rate are high, or when both encoding noise and drift rate are low. These are all regimes in which one stimulus (cue or probe) is encoded very noisily, while the other is encoded with high fidelity. To test this, we plan to parametrically degrade the cue and probe stimuli independently of each other. We predict that, in regimes with less peaky reaction time distributions (low drift rate), the DDP model will fit better when the probe uncertainty exceeds the cue uncertainty, and vice versa. In addition, we plan to test how the proactive index (AY – BX errors and RTs) relates to the time spent integrating probe or cue evidence, respectively.

References

[1] Miller, E.K. & Cohen, J.D. (2001). *Annu. Rev. Neurosci. 21, 167–202.* [2] Braver, T.S. et al. (2007) In *Variation in Working Memory* (Conway, A. et al., eds), pp. 76–106. [3] Braver, T.S. et al. (2009) *Proc. Natl. Acad. Sci. U.S.A. 106*, 7351–7356. [4] Bogacz, R. et al. (2006) *Psychol. Rev.* 113(4), pp. 700–765. [5] Busemeyer, J.R. & Townsend, J.T. (1993) *Psychol. Rev.*, 100(3), pp. 432-459. [6] Navarro, D.J. and Fuss, I. (2009) *Journal of Mathematical Psychology*, *53*, 222-230. [7] Fernandes, H.L. et al. (2014) *Cerebral Cortex 24*, 3232–3245; [8] Navarro, D., Pitt, M.A., & Myung, I. (2004). *Cognitive Psychology 49*, 47-84.

Coarse Q-Learning: Addressing the convergence problem when quantizing continuous state variables

Richard Dazeley
Federation Learning Agents Group (FLAG)
School of Engineering and Information Technology
Federation University Australia
Ballarat, VIC 3353
r.dazeley@federation.edu.au

Peter Vamplew
Federation Learning Agents Group (FLAG)
School of Engineering and Information Technology
Federation University Australia
Ballarat, VIC 3353
p.vamplew@federation.edu.au

Adam Bignold
Federation Learning Agents Group (FLAG)
School of Engineering and Information Technology
Federation University Australia
Ballarat, VIC 3353
a.bignold@federation.edu.au

Abstract

Value-based approaches to reinforcement learning (RL) maintain a value function that measures the long term utility of a state or state-action pair. A long standing issue in RL is how to create a finite representation in a continuous, and therefore infinite, state environment. The common approach is to use function approximators such as tile coding, memory or instance based methods. These provide some balance between generalisation, resolution, and storage, but converge slowly in multidimensional state environments. Another approach of quantizing state into lookup tables has been commonly regarded as highly problematic, due to large memory requirements and poor generalisation. In particular, attempting to reduce memory requirements and increase generalisation by using coarser quantization forms a non-Markovian system that does not converge. This paper investigates the problem in using quantized lookup tables and presents an extension to the Q-Learning algorithm, referred to as Coarse Q-Learning (C_{QL}), which resolves these issues. The presented algorithm will be shown to drastically reduce the memory requirements and increase generalisation by simulating the Markov property. In particular, this algorithm means the size of the input space is determined by the granularity required by the policy being learnt, rather than by the inadequacies of the learning algorithm or the nature of the state-reward dynamics of the environment. Importantly, the method presented solves the problem represented by the curse of dimensionality.

Keywords: Reinforcement Learning, Temporal Difference Learning, Continuous State, Quantized state, Function Approximation.

1 Introduction

Reinforcement Learning (RL) is a valuable tool for sequential decision making problems. However, while it works well with discrete states and actions, it can struggle with continuous state environments where an infinite state space must be represented within a finite amount of memory. The simplest approach is to quantize the state variables by partitioning them into a multidimensional grid. Each region, usually a hypercube, within the grid is then regarded as an atomic input or cell[1]. This approach is simple, but the resolution of quantization needs to be very fine to avoid the system becoming non-Markov and therefore never converging (Geramifard et al., 2013; Santamaría et al., 1997). However this exponentially increases the number of cells, and therefore the memory requirements, as well as vastly slowing learning. The failings of quantization have led researcher to seek other ways to reduce the size of the state representation while maintaining good performance. While a discussion of these is beyond the length of this paper they include more robust function approximators and state adaptation. Function approximators such as tile coding (Sutton, 1996) provide improved balance between generalisation, resolution, and storage over quantization (Santamaría et al., 1997), but converge slowly with multidimensional state environments. Other approaches have enabled temporal abstraction by limiting the points where decisions are made and allowing temporally-extended actions such as in options (Sutton et al., 1999). While these approaches have improved RL in continuous environments they still represent an attempt to find a work around, rather than a solution, to the problems of quantization.

This paper takes a fresh look at the quantization approach and investigates the cause of its behaviour under coarse partitioning. We propose a modification to Q-Learning (Watkins and Dayan, 1992), Coarse Q-Learning[2] (C_{QL}), which allows a much coarser quantization without exhibiting the divergent behaviour of Q-Learning. Results show that the coarsity of partitioning under C_{QL} is only limited by the policy being learnt. This eliminates the need to unneccesarily finely quantize domains and so reduces memory requirements, increases generalisation and speeds-up learning. The following section introduces a simple environment that illustrates Q-Learning's divergence under coarse quantization. Section 3 introduces C_{QL} and discuss the differences to Q-Learning. The next section then analyses C_{QL}'s performance and discusses why this algorithm converges successfully while Q-Learning diverges. It will also compare these algorithms using a non-smooth environment where C_{QL}'s potential for use within state adaptation is apparent.

2 Analysing Divergence in Quantized Continuous State Environments

In finite state environments a tabular representation can be used and Q-learning is guaranteed to converge (Geramifard et al., 2013; Santamaría et al., 1997). However, with continuous variables there are an infinite number of states, each requiring action-utility values. We can still use a tabular approach by quantizing variables into small discrete cells (Geramifard et al., 2013) where all states in each cell share the same state-action value. The challenge is determining how fine grained a representation needs to be to still converge to a solution. In domains with reasonably smooth state-action space (Kaelbling et al., 1996) discretising on the scale of the average step-size can yield a situation akin to a finite discrete problem in that an action will usually result in moving from one cell to the next in a single step. However this can represent a significant memory allocation and will significantly slow learning.

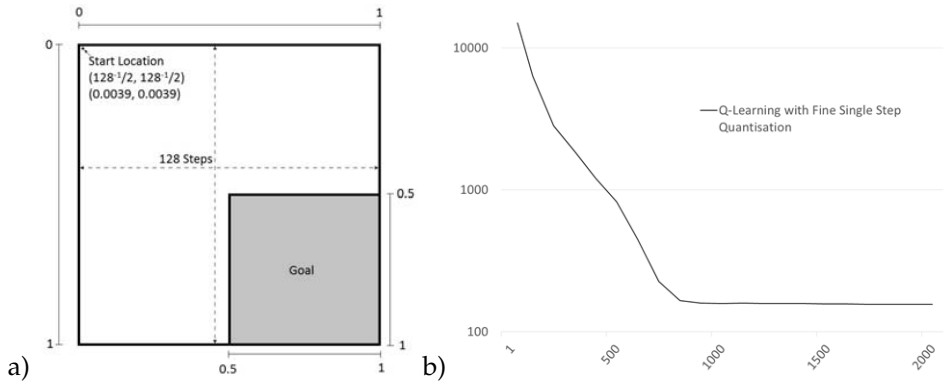


Figure 1: a) A simple environment where the agent must learn a path from the starting position to the goal. b) The performance of Q-Learning in terms of episode length when each cell is the size of a single step.

[1]Terminology varies with both the environment-variable and the quantized inputs often called states, while tile coding research refers to the quantized inputs as tiles. We will use state for the environment-variables' raw value, and cell for the quantized input.

[2]Late in the preparation of this paper we discovered that da Silva and Costa (2009) previously used this name, but the only similarity to our approach is the use of clamped actions. Their method did not work, but this is due to the incompleteness of the algorithm and the unstable nature of the state representation used in their evaluation. We address these issues in this paper.

The environment in Fig 1a illustrates the problem with quantization. Two continuous state variables representing an x, y coordinate in the range 0..1. Each time-step the agent can move a distance of 128^{-1} in any cardinal direction and receives a penalty of -0.1. The agent receives a penalty of -0.2 for colliding with an edge and a reward of 12.8 when the goal is achieved. Alhough this is an exceptionally simple task it requires $128_{(x)} \times 128_{(y)} \times 4_{(actions)}$ = 65536 values to accurately represent the environment using fine-grained quantization. Furthermore, when tested over ten trials Q-Learning took an average of 35881 episodes (stdev 44643) to converge to an optimal solution (Fig 1b). Quantizing in this fashion is clearly intractable in higher dimensional problems, as noted previously (Kaelbling et al., 1996; Geramifard et al., 2013; Santamaría et al., 1997). To reduce memory requirements and accelerate learning cells must cover more states. For instance, if cell size is doubled then the number of cells is reduced to 16384. Applying this two-step cell size to the above problem results in faster learning as it has less cell values to learn. However the agent never fully converges and starts behaving erratically. The non-Markovian environment and the resulting lack of convergence is clearer as the number of steps per cell is increased. Fig 2 illustrates Q-Learning's divergence as cell size is increased.

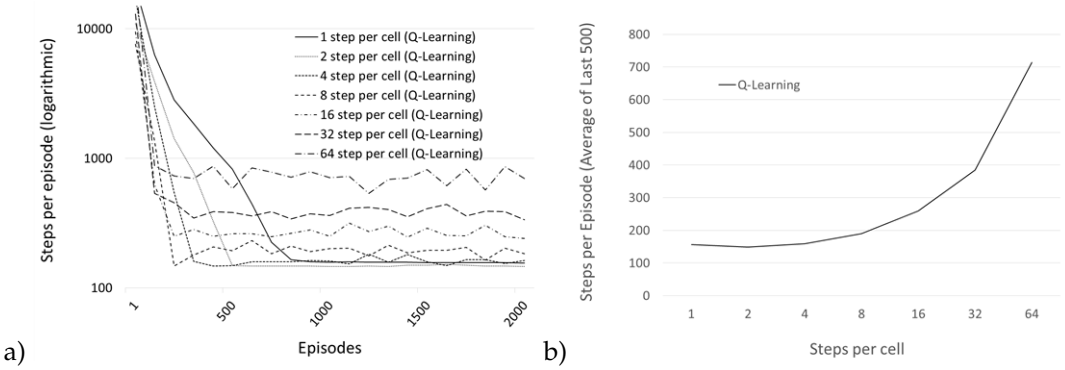


Figure 2: a) A moving average of the length of the last 100 episodes. With each increase in cell size the agent settles on a worse solution and behaves increasingly erratically. b) The average length of the last 500 episodes, illustrating that Q-Learning diverges as the number of steps in each cell increases.

The cause of the problem is that the agent must take multiple steps within each cell, which changes the problem to be non-Markovian (Geramifard et al., 2013), and therefore not guaranteed to converge using Q-Learning (Kaelbling et al., 1996). When multiple steps are taken in the same cell the agent must repeatedly select an action to perform. When close analysis is done of the cell-action values it can be seen that these decisions change over time. This is caused by the *self-referential learning loop* that has been created by the representation - the new cell-action value is based on the difference between the new cell's value and the original cell's value, plus any reward. When the new cell and the original cell are the same the difference is zero and the result is just the reward - hence the cell-action value will be driven up or down continuously and never converge. Furthermore, once a particular action has been taken its value will be changed and so when the agent must repeat the action-selection for this cell the choice may be different. This results in oscillating or cyclical behaviour, as shown in Fig 3.

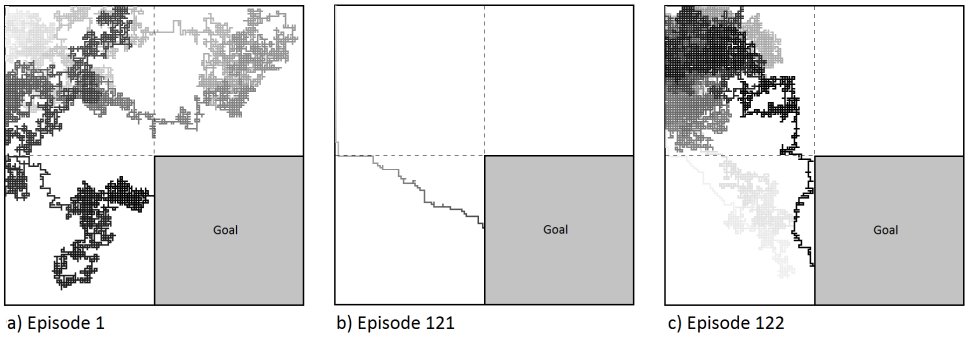


Figure 3: Shows Q-Learning behaviour in the environment from Fig 1 with 4 large cells. The path progresses from light at the start of an episode to dark at the end. a) shows the random walk at the start of training, b) shows that a decent policy has been learnt after 121 episodes, however, c) shows that policy is very unstable causing the agent to easily revert to an essentially random walk. This pattern of reverting back to a random walk repeats often.

3 Coarse Q-Learning

The aim for C_{QL} was to prevent the wandering seen in the last section, by modifying Q-Learning in order to simulate the Markov property by treating each cell $c \in \mathbb{C}$ in the way that a single state would be in a finite tabular state environment.

This is accomplished by using *clamped-actions* which are simply actions that, once selected, are repeated until a new cell is entered. When the agent enters a cell it selects a clamped-action and repeatedly executes it until the current cell is exited. Individual steps within that cell are treated as sub-steps of a single clamped-step by the agent. This is similar to the extended-actions used in options (Sutton et al 1999) except there is no underlying state as far as the agent is concerned. The use of clamping actions does raise some issues that will be discussed in the following subsections.

Cell Definition A cell, c, encapsulates one or more states, such that $c_1 \cup c_2 \cup \ldots \cup c_n = \mathbb{S}$, where $n \leq \mid \mathbb{S} \mid$. $\mathrm{C_{QL}}$ learns common action values for all states in a cell, so it is important that the optimal policy for each of the states contained in the cell is the same. Therefore cells should be formed such that: $if \pi^*(s) \neq \pi^*(s') then \nexists c_i$ where $s \in c_i$ and $s' \in c_i, \forall c_i \in \mathbb{C}$ This means the cardinality and structure of cells is tied to the optimal policy. Clearly, this policy is not known *a priori*, and so we anticipate that one main use of $\mathrm{C_{QL}}$ will be in conjunction with state adaptation to discover appropriate cells. This paper will only validate that it learns successfully and future work will investigate its utilisation with state adaptation. It should be noted here, that theoretically $\mathrm{C_{QL}}$ could be used in other generalisations such as tile coding which could break this cell selection rule but there should be at least some cells c_i where $s \in c_i$ and $s' \notin c_i$ or vice-versa.

Exiting Cells $\mathrm{C_{QL}}$ assumes that an agent repeatedly taking the same action will eventually exit the current cell. Where this does not hold then a means is required to detect this. One approach may be to identify a change in reward where an agent leaves a cell and then returns to the same cell (eg a wall collision) - this approach is used in this paper. Alternatively the agent may observe the continuous state and terminate the clamped-action if these values are not changing sufficiently, or it may time-out if more than a threshold number of steps have been taken without leaving the current cell.

Gathered Reward and Update Rule As the agent's choice of action is fixed until the current cell is exited, there is no need to update action values after each sub-step. Therefore $\mathrm{C_{QL}}$ sums the reward until the cell is exited, when the total reward is then applied - as if the cell was a single step in a finite tabular environment. As this is treated as a single step update, any discounting or λ attenuation will be less resulting in a larger update for earlier cells.

Algorithm 1 Coarse Q-Learning

Initialise $Q(s,a)$ arbitrarily and $e(s,a) \leftarrow 0, \forall s,a$
repeat for each episode:
  Initialise s,a; Initialise $t \leftarrow 0$
  repeat for each step of episode
    Take action a; observe r, s'; $t \leftarrow t + r$
    if $s' = s$ and not (stuck in cell) **then**
      $a' \leftarrow a$
    else
      Choose a' from s' using policy derived from Q (e.g. ε-greedy)
      $a^* \leftarrow \mathrm{argmax}_b Q(s',b)$ (if a' ties for the max, then $a^* \leftarrow a'$)
      $\delta \leftarrow t + \gamma Q(s',a^*) - Q(s,a)$; $t \leftarrow 0$
      $e(s,a) \leftarrow e(s,a) + 1$ // or $e(s,a) \leftarrow 1$
      for all s,a **do**
        $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
        if $a' = a^*$ **then**
          $e(s,a) \leftarrow \gamma\lambda e(s,a)$
        else
          $e(s,a) \leftarrow 0$
        end if
      end for
    end if
    $s \leftarrow s'; a \leftarrow a'$
  until s is terminal
until no more episodes

4 Empirical Evaluation of Coarse Q-Learning

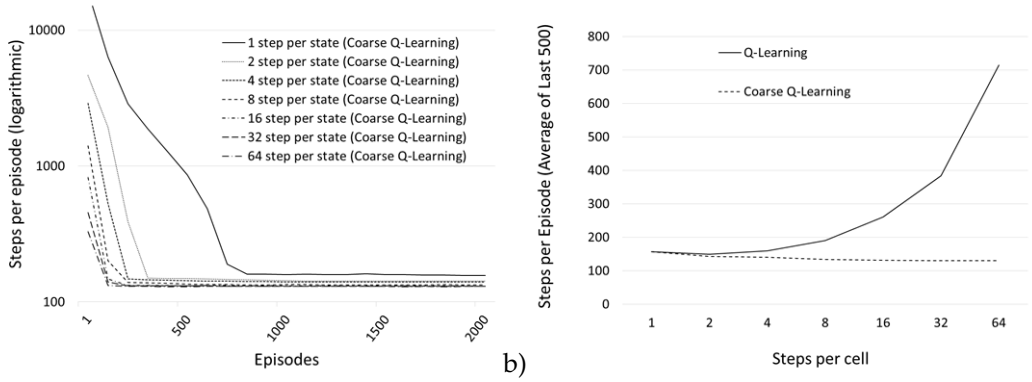


Figure 4: a) Results for $\mathrm{C_{QL}}$ over 2000 time steps with different cell sizes. b) Compares the final 500 episodes of Q-Learning and $\mathrm{C_{QL}}$.

Fig 4a shows how C_{QL} learns with different sized cells, while 4b compares it against Q-Learning. C_{QL} learns faster and does not show any signs of divergence as cell size increases. The results also show that if a fine quantization is used C_{QL} performs just like Q-Learning. Therefore, C_{QL} can be applied wherever Q-Learning is currently used without adversely affecting results, but if there is some coarsity in the state representation this algorithm will outperform Q-Learning.

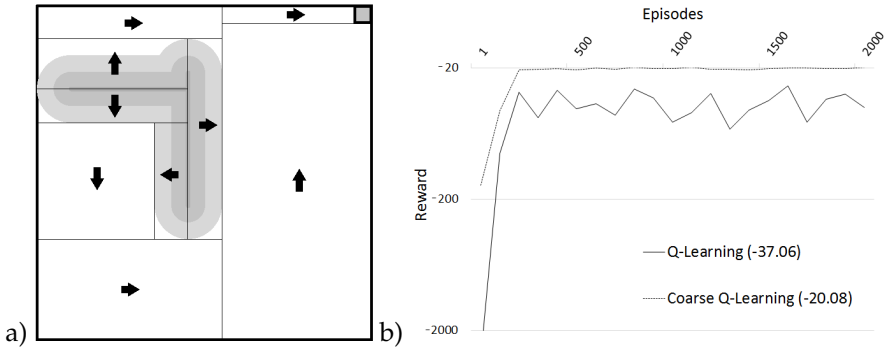


Figure 5: a) The expert created partitioning of PuddleWorld. b) Comparing the reward received of C_{QL} and Q-Learning on the expert-partitioning of PuddleWorld. The average of the last 500 episodes over ten runs is given in brackets.

Finally we look at how C_{QL} performs in a problem domain with a non-smooth state-action-reward - PuddleWorld (Boyan and Moore, 1995; Sutton, 1996). Due to the puddles a uniform coarse quanitization does not allow optimal performance, as cells may contain both puddle and free-space in which the optimal actions are different. A more effective coarse representation might be derived using state adaptation techniques (Whiteson et al., 2007; Pyeatt et al., 2001). We have simulated this by using an expert partitioning shown in Fig 5a which requires just 10 cells, far less than the 256 uniform cells we found were required for Q-learning to converge on this problem. In the results shown in Fig 5b, it can be seen that on this partitioning C_{QL} has far exceeded the performance of Q-Learning which never converges to a stable policy.

5 Conclusion

This paper took a fresh look at the quantization approach to generalisation. We identified the fundamental problem leading to divergence when Q-Learning is applied to coarsely quantized environments as the self-referential learning loop. This paper introduced Coarse Q-Learning, which uses clamped-actions and gathered rewards to prevent aimless wandering and to simulate the Markov property. C_{QL} was tested in a simple smooth environment and the non-smooth PuddleWorld bench mark and in both cases significantly outperformed Q-Learning. Most importantly it did not exhibit the same divergent behaviour clear in Q-Learning. The coarseness of partitioning under C_{QL} is only limited by the policy being learnt and so it can learn effectively without fine quantization, significantly simplifying the problem being learnt, reducing memory requirements, increasing generalisation and speeding-up learning.

References

Boyan, J. and A.W. Moore (1995), "Generalization in reinforcement learning: Safely approximating the value function." *Advances in neural information processing systems*, 369–376.

da Silva, V.F. and A.H.R. Costa (2009), "Compulsory Flow Q-Learning: an RL algorithm for robot navigation based on partial-policy and macro-states." *Journal of the Brazilian Computer Society*, 15, 65–75.

Geramifard, A, T J Walsh, S Tellex, G Chowdhary, N Roy, and J.P How (2013), "A tutorial on linear function approximators for dynamic programming and reinforcement learning." *Foundations and Trends® in Machine Learning*, 6, 375–451.

Kaelbling, L.P, M.L. Littman, and A.W. Moore (1996), "Reinforcement learning: A survey." *arXiv preprint cs/9605103*.

Pyeatt, L., A. Howe, et al. (2001), "Decision tree function approximation in reinforcement learning." In *the Third International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*, volume 2, 70–77.

Santamaría, J.C., R.S. Sutton, and A. Ram (1997), "Experiments with reinforcement learning in problems with continuous state and action spaces." *Adaptive behavior*, 6, 163–217.

Sutton, R.S. (1996), "Generalization in reinforcement learning: Successful examples using sparse coarse coding." *Advances in neural information processing systems*, 1038–1044.

Sutton, R.S., D. Precup, and S. Singh (1999), "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." *Artificial intelligence*, 112, 181–211.

Watkins, Christopher JCH and Peter Dayan (1992), "Q-learning." *Machine learning*, 8, 279–292.

Whiteson, S., M. Taylor, P. Stone, et al. (2007), *Adaptive tile coding for value function approximation*. Computer Science Department, University of Texas at Austin.

Reward Shaping by Demonstration

Halit Bener Suay
Worcester Polytechnic Institute
benersuay@wpi.edu

Tim Brys
Vrije Universiteit Brussel
timbrys@vub.ac.be

Matthew E. Taylor
Washington State University
taylorm@eecs.wsu.edu

Sonia Chernova
Worcester Polytechnic Institute
soniac@wpi.edu

Abstract

Potential-based reward shaping is a theoretically sound way of incorporating prior knowledge in a reinforcement learning setting. While providing flexibility for choosing the potential function, under certain conditions this method guarantees the convergence of the final policy, regardless of the properties of the potential function. However, this flexibility of choice may cause confusion when making a design decision for a specific domain, as the number of possible candidates for a potential function can be overwhelming. Moreover, the potential function either can be manually designed, to bias the behavior of the learner, or can be recovered from prior knowledge, e.g. from human demonstrations. In this paper we investigate the efficacy of two different methods of using a potential function recovered from human demonstrations. Our first approach uses a mixture of Gaussian distributions generated by samples collected during demonstrations (Gaussian-Shaping), and the second approach uses a reward function recovered from demonstrations with Relative Entropy Inverse Reinforcement Learning (RE-IRL-Shaping). We present our findings in Cart-Pole, Mountain Car, and Puddle World domains. Our results show that Gaussian-Shaping can provide an efficient reward heuristic, accelerating learning through its ability to capture local information, and RE-IRL-Shaping can be more resilient to bad demonstrations. We report a brief analysis of our findings and we aim to provide a future reference for reinforcement learning agent designers who consider using reward shaping by human demonstrations.

Keywords: Reinforcement Learning, Inverse Reinforcement Learning, Reward Shaping, Learning from Demonstration.

Acknowledgements

Halit Bener Suay's work is supported by the Office of Naval Research award N000141410795.

Tim Brys is funded by a PhD scholarship of the Research Foundation Flanders (FWO).

1 Introduction

Incorporating prior knowledge in Reinforcement Learning (RL) in a domain-independent way is an open research question. While there is not a single, domain-generic, *best* way, of incorporating prior knowledge in RL, some options include: using a model of the environment dynamics, having an initial policy, a value function, a reward function, or initial state values [1, 2].

In this paper we explore two new options for incorporating human demonstrations into the reward signal of an RL agent [3]. The first approach we present, *Gaussian-Shaping* [4], takes advantage of the demonstrations locally within the state space and allows demonstrations of different lengths in order to generate multiple Gaussian distributions for reward shaping with a potential function based on a similarity metric (i.e., the distance to the closest demonstrated state-action pair). The second approach, *Relative Entropy Inverse Reinforcement Learning-Shaping* (RE-IRL-Shaping), is based on feature matching and it uses a set of demonstrated trajectories in order to infer a potential function that generalizes over the entire state space for shaping. Feature matching is one of the main approaches to solve the inverse reinforcement learning (IRL) problem. In this approach an IRL algorithm tries to find a set of parameters for a reward function, in order to maximize the reward for experiences with matching feature counts to a demonstration set. RE-IRL algorithm is a model-free algorithm that finds reward feature weights by minimizing the relative entropy between a set of demonstrated features, and features along trajectories that are sampled from a demonstrator's internal policy modeled as a probability distribution. A common property of both of our approaches is that the potential function serves just as a heuristic reward input in the RL setting. An important implication is that when learning just a reward heuristic, a learner does not need to assume that the demonstrator is an expert, or that the outcome of the inferred data is optimal by any metric. This is because the environment reward acts as a consistent, true reward signal that the agent can rely on. Therefore as long as the demonstrations are not deliberately malicious to hinder learning, we can drop the requirement that the demonstrations originate from an expert demonstrator. Note that, unlike our work, most state-of-the-art learning from demonstration algorithms do make the assumption that the demonstrator is an expert for learning a task [5]. We investigate the learning performance of our agents in three commonly used simulated domains: Cart-Pole, Mountain Car, and Puddle World. We aim to help future RL solver designers make more informed design decisions through comparative analysis of the presented methods.

To give an insight into our motivation, we first look at how prior work has incorporated human input in the RL setting. Knox and Stone introduced the TAMER framework where an RL agent receives feedback only from a human in order to model the internal reward signal of the human. Although agents in the TAMER framework do incorporate human input in the RL setting, the framework omits the environment signal altogether, which contrasts with our approach [6].

In a follow up work, Knox and Stone designed a comparative based on analysis of eight different techniques for combining the modeled human reward signal with environment reward. Approaches they introduced varied from using the modeled human reward as a potential function for shaping the environment reward, to simply adding an extra action choice that maximizes the modeled human reward [7]. The general finding of the study is that the combination of a modeled human reward function with a RL agent outperforms the RL agent, and the agent that maximizes the modeled human reward alone. The general idea of using multiple reward functions is similar to our approach in spirit, however our approach is not interactive. Instead we first learn reward functions based on a demonstration data-set. Moreover instead of comparing different techniques for *merging the environment reward* with the human reward signal, we analyze *the use of two different shaping rewards*.

Griffith et al. use simple human feedback such as *right* or *wrong* for shaping a policy in interactive reinforcement learning setting [8]. The approach they present interprets the human feedback as a comment on the optimality of actions. In our work we shape the reward function, and instead of treating the human input as the ground truth, both of our approaches use demonstrations to recover a reward heuristic.

To the best of our knowledge, our choice of potential functions for reward shaping, both using a mixture of Gaussians, and RE-IRL has not been proposed, or studied before. Next, we talk about the two different potential functions we used for reward shaping, the functions' effects on the performance curve of a Q-Learner agent, and our interpretation of the results.

2 Reward Shaping by Demonstration

In this paper we focus on recording the trajectory (s_t, a_t) of an agent throughout multiple episodes of teaching, and using the recorded trajectories for inferring a shaping function. When available, if we rely on the environment reward as a consistent source of reward, we can drop the assumption that the human input has to be optimal. We can shape the standard reward function using a potential function generated based on demonstrations. Even though the demonstrations should still be non-malicious (i.e. not target toward purposefully confusing the learner), reward shaping supplies an agent with a biased, and hopefully more informative reward input. Our baseline agent is a Q-Learning agent, and in both of our methods, the only addition we make is for the shaping of the reward function of a standard Q-Learner. Watkins defines the standard update for a Q-function [3] as

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s,a,s') + \gamma \arg\max_{a'} Q(s',a') - Q(s,a)) \quad (1)$$

In our methods, instead of using just a reward function $R(s,a)$ we add a shaping reward $F(s,a,s',a')$ and use a shaped reward $R' = R+F$. We compute the shaping reward F based on a potential function $\Phi(s,a)$ [9, 10]. Next we present how we infer the potential function $\Phi(s,a)$, and the shaping reward $F(s,a,s',a')$ from human demonstrations. We would like to highlight that neither of our techniques require the model of the environment, but only human interaction with the system for collecting demonstration data.

2.1 Using Local Information with Gaussian Distributions

Here we introduce our first method for inferring a potential function Φ using demonstration data, based on sample (i.e. state) similarity given a discrete action a. We show i) how we decide what constitutes a *similar sample*, ii) how we pick the most similar sample within a demonstration set, and once we do, iii) how we use the similarity as a heuristic for reward shaping. We use the following set of equations:

$$\Sigma = \sigma I \tag{2}$$

$$g(s, s^d, \Sigma) = e^{(-\frac{1}{2}(s-s^d)^T \Sigma^{-1}(s-s^d))} \tag{3}$$

$$\Phi(s,a) = \max_{(s^d,a)} g(s, s^d, \Sigma) \tag{4}$$

$$F(s,a,s',a') = \gamma\Phi(s',a') - \Phi(s,a) \tag{5}$$

where, s is the state that is being experienced by the agent at a given time, s^d is one of the states demonstrated which we iterate through, I is an identity matrix, and Σ is the covariance matrix that defines the reach of influence of demonstrated state-action pairs. Σ is domain specific and we tune σ manually, however it is possible to learn metrics, such as Σ, autonomously for RL agents [11]. The intuition for tuning Σ is to set it to larger values for higher dimensional state-spaces. Eq. 3 is a multivariate Gaussian distribution, which outputs the value of a similarity metric that varies between 0 and 1 depending on the distance between the current state and the closest demonstrated sample within the set of demonstrations. Eq. 4 is the potential function for reward shaping, based on which we compute our shaping reward using Eq. 5 [10].

The idea behind our algorithm is simple: once we decide the spread of demonstrated samples and tune σ (Eq. 2), we iterate through demonstrated samples using Eq. 3, and assign the maximum similarity value as the value of the potential for the state (Eq. 4). We compute the similarity metric both for the current state s' and the previous state s of the agent, and compute the difference using Eq. 5, where γ is the discount factor. For a more detailed description and analysis of this approach we refer the readers to Brys et al. [4].

Eq. 3 is the reason why we call this approach *local*. Given an action and a set of demonstrations, encounters with states that are similar to demonstrated states result in high potential for reward shaping. If the agent transitions from a state s that is not similar to any of the demonstrated states, into a state s' that is very similar to one of the demonstrated states, Eq. 5 returns a positive shaping reward indicating that the agent took an action in a *good* direction. In the opposite case, Eq. 5 returns a negative shaping reward.

Even though using sample similarity in this fashion helps to guide the agent toward demonstrated states, this approach imposes a constraint on the quality of the demonstrations themselves. Since currently we do not incorporate the quality of a given sample within our algorithm (i.e. good demonstration vs. bad demonstration), similarity to all demonstrated states is considered to be equally good. That is, if the agent has been demonstrated an action in a state similar to its current state, this can only mean that the action is valuable in that current state. Consequently, if the set of demonstrations includes undesired data points (intentional or unintentional), when the agent experiences states similar to these *undesired* data points, it is possible that the resulting shaping reward will be positive, which would degrade the agent's learning performance. We recognize the importance of automated demonstration quality assessment. One idea for assessing the demonstration quality is to analyze the statistics of the features which may give an idea about the consistency of the data, however we leave this idea for future work, since it deserves to be the main focus of a separate analysis.

2.2 Using RE-IRL As a Heuristic for Reward Shaping

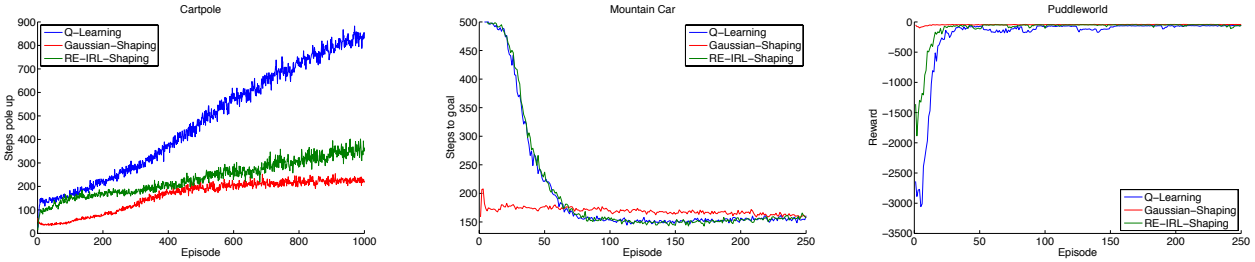
Our second method for inferring a potential function Φ from demonstration data is based on Relative Entropy Inverse Reinforcement Learning (RE-IRL), a model-free inverse reinforcement learning technique introduced by Boularias et al. [12]. The input for RE-IRL is a set of demonstrations which are trajectories of equal length, and the output is a set of reward feature weights ω required to compute a reward value for a given state s. We show i) how we compute a reward using a linear combination of learned feature weights, and ii) how we use the reward as the potential function for reward shaping. The set of equations we use for our approach are:

$$R_{IRL}(s,a) = \sum_{i=0}^{n} \omega_i f_i(s) \tag{6}$$

$$\Phi(s) = R_{IRL}(s,a) \tag{7}$$

$$F(s,a,s',a') = \gamma\Phi(s') - \Phi(s) \tag{8}$$

where, f_i is ith reward feature, ω_i is ith feature weight, s' is the state vector for the current state of the agent, and s is the previous state vector.



(a) (b) (c)

Figure 1: Learning performance results in (a) Cart-Pole, (b) Mountain Car, and (c) Puddle World domains.

Coeff.	Cart-Pole	Mountain Car	Puddle World
α	$\frac{0.25}{16}$	$\frac{0.1}{14}$	$\frac{0.2}{10}$
γ	1.0	1.0	1.0
ϵ	0.05	0.1	0.05
λ	0.25	0.95	0.95
σ	0.2	0.1	0.1

(a)

variable	ω_i
$cart_x$	0.4245
$cart_{\dot{x}}$	0.56805
$pole_\theta$	0.33986
$pole_{\dot{\theta}}$	0.61774

(b)

variable	ω_i
x	-0.999
$\dot{x}$	0.001

(c)

variable	ω_i
x	0.76
y	0.6497

(d)

Table 1: Learning coefficients we used for our experiments (a). Reward feature weights in (b) Cart-Pole, (c) Mountain Car, and (d) Puddle World domains. x and y are position variables; $\dot{x}$ is velocity; θ is angle and $\dot{\theta}$ is angular velocity.

Our approach has two parts: first we learn a reward function by recovering reward feature weights based on a set of demonstrations, and then we use the learned reward function as a potential function for reward shaping. In order to obtain reward feature weights, RE-IRL computes the average feature values (for each feature) over a set of demonstrated trajectories, and uses importance sampling for estimating feature weights in order to iteratively drive a gradient to zero. The gradient is simply the difference between the average (demonstrated) feature values and sampled feature values. For further details of the RE-IRL we refer our readers to [12]. Instead of using the learned function as a reward function (Eq. 6), we define a potential function (Eq. 7) for shaping reward (Eq. 8). This algorithmic setup incorporates a function we learn from human demonstrations into autonomous learning as a shaping reward (or heuristic).

The weights we recover for each reward feature indicate how each feature value affects the reward output. For instance a feature can positively, or negatively dominate others in terms of reward output and understanding each feature's contribution in the reward output can be informative. Moreover RE-IRL computes feature weights using the entire trajectory space and set of demonstrations. As a consequence of average feature matching, this approach is robust against small changes in demonstrations. This *global* use of the demonstration data is in contrast with the previous approach we introduced. If the agent transitions into a state where, the linear combination of the reward features is higher than a previous state, this transition yields a positive shaping reward.

Although we take advantage of the demonstration data on average, in this approach we are limited with the linear combination of the reward features for computing the potential function. This is an important factor to consider when choosing which reward features to use. Even though the reward features can be arbitrarily engineered to design an ad-hoc solution depending on the domain, here for simplicity and compatibility with our previous approach, we chose to use the state variables as reward feature in each domain. We acknowledge that as a future work we need a thorough investigation for choosing the set of reward features.

2.3 Experiments

We tested our approach in three domains using 10 demonstrated trajectories for each domain. The length of demonstrated trajectories respectively were 156, 182, and 39 respectively for Cart-Pole, Mountain Car, and Puddle World domains. In order to keep the demonstration lengths equal, for shorter demonstrations, we appended the final state as an absorbing terminal state at the end of the demonstration. The demonstrations we recorded begin at the initial state and end at the goal (for Mountain Car and Puddle World) *or* the failure state (for Cart-Pole) of the agent. The demonstrator is the first author who has experience with all three domains. The demonstrator made the demonstrations for reaching the goal as quickly as possible in Mountain Car and Puddle World, and for keeping the pole in balance as long as possible for the Cart-Pole domain. Table 1a shows the RL coefficients we used for our experiments.

Table 1b - 1d list the feature weights we use for RE-IRL-Shaping, which we obtained from our set of demonstrations. We can see that in Cart-Pole domain, positive values of state variables result in a positive potential. Although these values are based on demonstrations, one can argue that encouraging the agent toward positive states is not optimal in Cart-Pole. RE-IRL-Shaping performed better than our Gaussian-Shaping algorithm (Fig. 1a). We conclude that this difference

originates from the way our two techniques take advantage of the demonstrations. However, in Cart-Pole the standard Q-Learning agent outperformed both of our approaches.

For RE-IRL-Shaping agent, in the Mountain Car, the position of the car is negatively weighted, and even though positive velocities contribute positively, the effect of the velocity of the car is highly dampened in comparison with the position of the car. This is because the demonstration data includes left and right swings of the car until the car reaches the goal state. The initial state in the domain is $s_0 = [-0.5, 0.0]$, and the shaping reward is positive as long as the car goes to the left. This is the only way for the car to speed up fast enough, so it can climb the hill to reach its goal. This set of weights provides an informative heuristic for the RE-IRL-Shaping agent however its performance is not any better than the standard Q-Learner.

In the Puddle World domain, weights for both state variables (x and y position) yield a high potential as the RE-IRL-Shaping agent goes to the right and up in the world. This aligns well with the purpose of the agent since the goal is on the upper right corner of the world, and we can see the positive effect of this potential function on the learning performance of RE-IRL-Shaping agent especially within the first 30 episodes. This algorithm outperforms the standard Q-Learner, even though Gaussian-Shaping algorithm shows the best overall learning performance.

As shown in Fig. 1b and 1c, the Gaussian-Shaping agent outperforms the standard Q-Learning agent, and RE-IRL-Shaping agent in Mountain Car and Puddle World domains. In both domains, the agents start from an initial *non-goal* state, and the common purpose is to keep moving until reaching a goal state. We use demonstration data not only to help guide the agents toward desired states, but also to initialize the Q-function using Eq. 4, with $Q_0(s, a) = \Phi(s, a)$ as described by Wiewiora et al. [10]. The results show a remarkable boost in the learning performance of Gaussian-Shaping agent in these two domains, starting from the very first episode.

On the other hand, in the Cart-Pole, agents start in a desired equilibrium state, and the purpose is to try to stay close to that initial state. This fundamental difference in the purpose of the task, the level of difficulty of the task for a demonstrator, and the failure (or borderline failure) states that are saved in demonstrations, affect the learning performance of the Gaussian-Shaping agent (Fig. 1a). The performance of the agent suffers from samples that are close to failure states, which highlights the importance of the content of the demonstrations.

3 Conclusion

With this work we present two different potential functions for reward shaping: a function based on similarity of the samples to demonstrated states, and a function learned with inverse reinforcement learning. Both potential functions enable us incorporate prior knowledge in RL. Our analysis in three standard RL domains shows that our sample similarity approach speeds up learning remarkably. However, it is sensitive to states that may be less optimal that are included in the demonstration data-set. On the other hand, even though with the current choice of reward features, our inverse reinforcement learning approach is less effective in improving the learning performance, when the majority of the demonstrations are in agreement, this approach is comparatively more robust against small disturbances in the demonstration data. We leave addressing the noise sensitivity, and reward feature selection issues for future work.

References

[1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[2] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *IJRR*, 2013.

[3] C. J. C. H. Watkins, *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[4] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, pp. 469–483, May 2009.

[6] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The tamer framework," in *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16, ACM, 2009.

[7] W. B. Knox and P. Stone, "Combining manual feedback with subsequent mdp reward signals for reinforcement learning," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2010.

[8] S. Griffith, K. Subramarian, J. Scholz, C. Isbell, and A. Thomaz, "Policy Shaping: Integrating Human Feedback with Reinforcement Learning," in *Advances in Neural Information Processing Systems*, pp. 2625–2633, 2013.

[9] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *In Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, Morgan Kaufmann, 1999.

[10] E. Wiewiora, G. W. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pp. 792–799, 2003.

[11] M. E. Taylor, B. Kulis, and F. Sha, "Metric learning for reinforcement learning agents," in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, (Richland, SC), pp. 777–784, International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[12] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pp. 182–189, 2011.

Ensembles of Shapings

Tim Brys
Vrije Universiteit Brussel
Brussels, Belgium
timbrys@vub.ac.be

Anna Harutyunyan
Vrije Universiteit Brussel
Brussels, Belgium
aharutyu@vub.ac.be

Matthew E. Taylor
Washington State University
Pullman, WA
taylorm@eecs.wsu.edu

Ann Nowé
Vrije Universiteit Brussel
Brussels, Belgium
anowe@vub.ac.be

Abstract

Many reinforcement learning algorithms try to solve a problem from scratch, i.e., without *a priori* knowledge. This works for small and simple problems, but quickly becomes impractical as problems of growing complexity are tackled. The reward function with which the agent evaluates its behaviour often is sparse and uninformative, which leads to the agent requiring large amounts of exploration before feedback is discovered and good behaviour can be generated. Reward shaping is one approach to address this problem, by enriching the reward signal with extra intermediate rewards, often of a heuristic nature. These intermediate rewards may be derived from expert knowledge, knowledge transferred from a previous task, demonstrations provided to the agent, etc. In many domains, multiple such pieces of knowledge are available, and could all potentially benefit the agent during its learning process. We investigate the use of ensemble techniques to automatically combine these various sources of information, helping the agent learn faster than with any of the individual pieces of information alone. We empirically show that the use of such ensembles alleviates two tuning problems: (1) the problem of selecting which (combination of) heuristic knowledge to use, and (2) the problem of tuning the scaling of this information as it is injected in the original reward function. We show that ensembles are both robust against bad information and bad scalings.

Keywords: Reward Shaping; Ensembles

Acknowledgements

Tim Brys is funded by a Ph.D grant of the Research Foundation Flanders (FWO). Anna Harutyunyan is supported by the IWT-SBO project MIRAD (grant nr. 120057). This work was supported in part by NSF IIS-1149917.

1 Motivation

With many reinforcement learning algorithms taking a *tabula rasa* approach, their sample complexity is often prohibitively high to be useful in realistic settings. In other words, they require too many experiences, too much 'trial-and-error,' before reaching a desirable level of performance. Imagine a task where the agent only receives positive reward after a very specific, complex sequence of actions has been executed (e.g., the 'combination lock' problem [1]). If the goal of the task is to execute this sequence of actions, then the reward function perfectly encodes this task. But, due to its sparsity, it will also likely result in very slow learning. A lot of research has therefore focused on speeding up these reinforcement learning algorithms by steering their exploration based on expert knowledge [2], knowledge transferred from previous tasks [3, 4], provided demonstrations [5, 6], human advice [7, 8, 9], abstract knowledge learned during learning [10], etc. In many cases, one has several such pieces of information available, e.g., several heuristic rules can be devised, or multiple source tasks are available to transfer from, etc. Problems a system designer is then faced with are (1) how to include this knowledge in the learning process, and (2) how to combine the various pieces of knowledge in an optimal way. More often than not, the system designer would start his own trial-and-error process of trying out combinations and tuning their parameters. This tuning process often requires many more experiences than are gained in the end by using the best performing combination. In reality, we would like an off-the-shelf solution that we can supply with different forms of information, and that can combine these automatically in a near-optimal way.

Our contribution towards this goal consists of injecting the information in the learning process through an approach called reward shaping, and using ensemble techniques to automatically and robustly combine the various pieces of information supplied.

2 Reward Shaping

Recall the example above, where the environment's reward is only positive when the required sequence of actions has been executed. If we can find a way to provide positive feedback for each step in this sequence of actions, the task will become easily learnable for the agent, as its behaviour is reinforced at every step. Of course, typically we do not know the solution beforehand, and can only provide information of a heuristic nature, i.e., rules of thumb that provide general guidelines, but are not perfect in every situation.

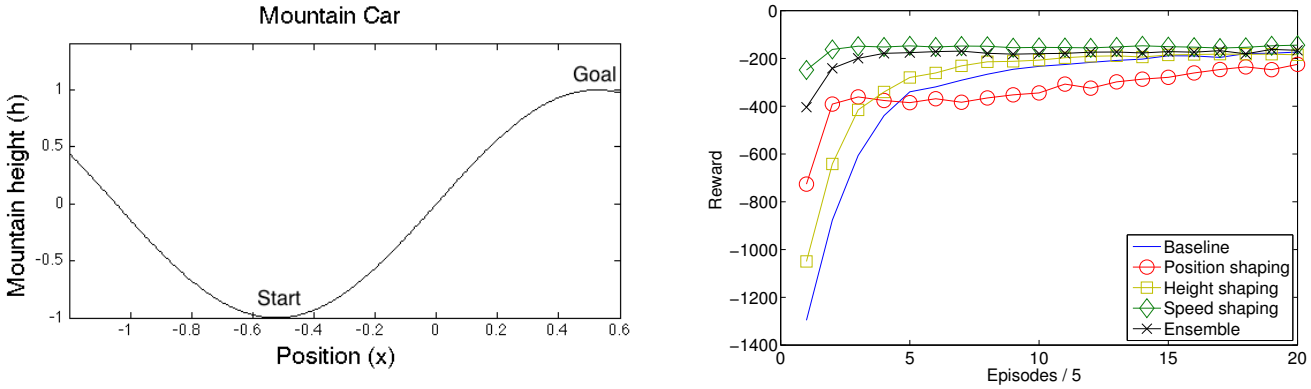
The idea behind reward shaping is to harness such information to enrich the environment's sparser reward and thus provide faster, more informative feedback for the agent's behaviour. The agent is supplied with an extra reward signal F that is added to the environment's reward R, making the agent learn on the composite signal $R_F = R + F$. Since the agent's goal is defined by the reward function (solving the task optimally means finding a description of behaviour, i.e., a policy, that achieves the maximum accumulated reward in expectation), changing the reward signal may actually change the task. Ng et al. [2] proved that the only sound way to modify the reward, while guaranteeing that the task's optimal policy does not change, is through potential-based shaping. That is, define a potential function Φ over the state space, and define F as the difference between the potential of states s' and s, given observed transition (s, a, s'):

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s)$$

This formulation preserves the total order over policies, and therefore also the optimality of policies. It has been successfully used to facilitate solving of such complex tasks as RoboCup TakeAway [11], StarCraft [12], Mario [13], helicopter flight [14, 15], etc.

The intuitition behind defining Φ is that states with high potential will be desirable to the agent, i.e., it will be encouraged to explore such states. A good potential function should therefore yield higher and higher potentials as the agent gets closer and closer to states that are desirable with respect to the base reward, thus quickly leading the agent to optimal behaviour. Again, in absence of knowledge of the full solution, we can only use heuristic information when defining Φ. Consider Mountain Car [16], a problem where an underpowered car, starting in the valley between two hills, needs to learn how to drive to the top of one of the hills by driving up and down the opposing hills, thereby building up momentum until it can finally reach the goal. See Figure 1 (a) for a visual representation of the problem. A first heuristic one could devise is to encourage the car to gain height: $\Phi(s) = height(s)$. Since the goal location is at the top of a hill, this makes sense. But, as the car needs to build up momentum by driving up and down the two hills, in many situations, the car should actually choose to go down instead of trying (and failing) to get further up the hill. Thus, this rule is not perfect. Another heuristic that can be devised is to encourage increasing speed: $\Phi(s) = speed(s)$. This also makes sense, as the underpowered car needs some initial speed to climb up the hill. But, again, the car needs to drive up the hills many times, each time slowing down in the process, so this heuristic is not perfect either. Basically, the car is constantly trading potential and kinetic energy until it can reach the goal. While both heuristics could be useful for helping the agent solve the task, it is unclear how we could optimally combine them without going through an extensive tuning and engineering phase.

To overcome this problem, we propose an ensemble approach to reward shaping with multiple heuristics that automates the process of combining them.



(a) (b)

Figure 1: (a) A visual representation of Mountain Car. (b) An ensemble of shapings in Mountain Car. The ensemble of shapings approximates the performance of the best shaping, which was unknown *a priori*.

3 Ensembles of Shapings

Ensemble techniques were developed to combine multiple 'weak' decision making algorithms into a stronger 'super' decision maker, aiming to outperform any of the constituting components. We propose to apply this idea to combine many 'weak' or suboptimal heuristics in reinforcement learning.

To do so, we first create multiple copies of the reward signal, each injected with a different potential-based reward shaping function. Given scalar reward function R and potential-based shaping functions F_1 through F_m, we construct a multi-objective reward function $\mathbf{R} = [R + F_1, \ldots, R + F_m]$ [13, 17, 18]. This process is called *multi-objectivization*. Each of the individual signals encodes a different piece of heuristic information and could be used on its own to solve the task, but we posit that creating an ensemble using these signals can help to solve the task faster by combining the different heuristics automatically.

An ensemble is then created by having m off-policy learning algorithms learn in parallel on the same experiences, each evaluating the behaviour according to one of the different enriched reward signals. The Horde [19] architecture is well-suited for this purpose given its off-policy convergence guarantees with linear function approximation and its computational efficiency, although it does place restrictions on the behaviour policy. In practice, an ensemble of Q-learners may work equally well, although it lacks convergence guarantees as strong as Horde.

An ensemble policy π is derived by combining each component's preferences:

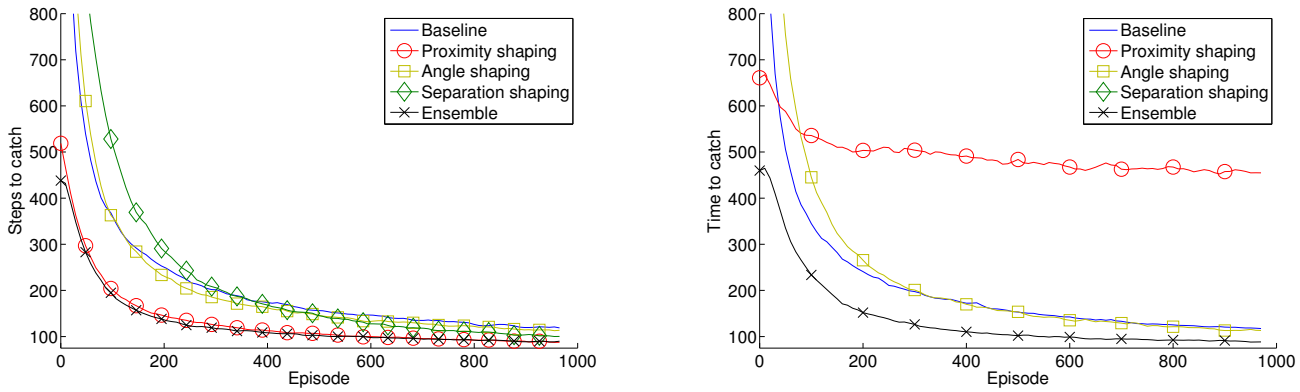
$$\pi(s) = \arg\max_a \sum_i^m p_i(s, a)$$

These preferences could be simple votes or rankings [20] or more complex dynamic, confidence-based preferences [17]. The preferences of each ensemble component will be biased by the heuristic that component is using, and employing a combination mechanism like majority voting ensures that the ensemble action will be their common denominator. Thus, even though heuristics do not apply in every situation, they can compensate for each other's suboptimality.

4 Mountain Car

In the Mountain Car task, discussed above, a learning agent receives a negative reward for every step taken, and this sequence of negative rewards only stops when he arrives at the goal location. Therefore, the optimal policy, maximizing the accumulated rewards, reaches the goal in a minimum number of steps. The reward function itself is very uninformative, as it does not provide any gradient information towards the goal. We suggested using height and speed as heuristics to help the learning agent find such behaviour faster. A third heuristic we will investigate is encouraging the agent to move to the right, since the goal is located at the far right of the world.

In Figure 1, we compare the performance of the $GQ(\lambda)$ reinforcement learning algorithm [21] learning to solve Mountain Car without shaping, with a single one of the three proposed shaping functions, and with an ensemble of shapings encoded in the Horde architecture. Majority voting is employed as the mechanism to derive a policy from the ensemble. We observe that the speed shaping is the most useful of the three heuristics when used on their own, and, while this knowledge was not available *a priori* , the ensemble automatically approximates this best performance.



Tuned scalings Original scalings

Figure 2: An ensemble of shapings in Predator-Prey. The scaling of the individual shapings has a large impact on performance, while the ensemble is unaffected.

5 Predator-Prey

We perform a similar experiment in the multi-agent Predator-Prey or Pursuit domain [22]. Two predators must learn how to coordinate in order to catch a fleeing prey, and only receive a positive reward when the prey is caught by at least one of the predators. Three heuristics we propose that could help are: (1) encouraging proximity to the prey, (2) encircling of the prey, and (3) separation between the predators. Figure 2 (a) compares the performance between standard $Q(\lambda)$-learning without shaping, with one of the three shapings, and an ensemble that uses a confidence measure to combine the three shapings [17]. The ensemble automatically, without prior knowledge, outperforms the best shaping alone (about 15% improvement in initial performance), even though two of the three shapings on their own are ineffective or detrimental to performance compared with the baseline.

An issue with the previous experiments is that we needed to tune the magnitudes of the shapings in order to provide the best performance. Of course, all this tuning is counterproductive, since our goal is to minimize the sample complexity of reinforcement algorithms, whereas tuning requires extra samples to select the best performing variants. Now, we argue that ensembles of shapings are not only robust with respect to the quality of the different heuristics provided, but also to their scalings. Figure 2 (b) shows the results of the same Predator-Prey experiment as in the (a) part of that figure, except that we simply left the magnitudes of the shapings as they were pre-tuning. In this case, every shaping on its own is detrimental to performance, yet the ensemble performs similar to the situation with the tuned scalings.

In other work, we have taken this one step further, by including multiple versions of the same heuristic in the ensemble, each version differently scaled [23]. In that paper, we present experiments in Mountain Car and Cart Pole that show how such an ensemble completely removes the need for tuning, automatically approximating the fastest possible learning given several shapings and arbitrary ranges of scalings. That is the first approach to reward shaping that truly removes the need for tuning.

6 Conclusions and Future Work

Reward shaping is a useful tool to incorporate prior knowledge to help a reinforcement learning agent reduce the number of experiences required to reach a desirable level of performance. Yet, in order for the shaping to be successful, there is usually some tuning necessary with respect to what knowledge to include, and how to scale the magnitude of the shaping compared to the base reward signal's magnitude. The number of extra experiences required for this tuning phase will typically be much higher than what is gained in the end by applying the best shaping.

In this work, we have investigated the use of shaping ensembles to remove this need for tuning, proposing an off-the-shelf solution that automatically can combine many different pieces of information. Ideally, a system designer can now create a number of shaping functions based on the information he has (even creating multiple shapings with the same piece of knowledge, but differently scaled) and combine them in an ensemble that automatically uses this information in a good, if not best, way.

As we discussed before, reward shaping can be used to encode things other than heuristic expert knowledge. Therefore, ensembles could for example be used to achieve multi-task transfer, assuming each source task will contribute different information to the target task, or to incorporate demonstrations given by different experts, assuming different experts'

demonstrations are significantly different. The ultimate goal is to demonstrate how an ensemble provided with a number of these types of information can allow a reinforcement learning agent to solve a complex practical application, such as a robotics manipulation task.

References

[1] B. R. Leffler, M. L. Littman, and T. Edmunds, "Efficient reinforcement learning with relocatable action models," in *AAAI*, vol. 7, pp. 572–577, 2007.

[2] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, vol. 99, pp. 278–287, 1999.

[3] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *The Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[4] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Policy transfer using reward shaping," in *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2015.

[5] M. E. Taylor, H. B. Suay, and S. Chernova, "Integrating reinforcement learning with human demonstrations of varying ability," in *The 10th International Conference on Autonomous Agents and Multiagent Systems*, pp. 617–624, 2011.

[6] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[7] W. B. Knox and P. Stone, "Combining manual feedback with subsequent mdp reward signals for reinforcement learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pp. 5–12, 2010.

[8] R. Loftin, J. MacGlashan, B. Peng, M. E. Taylor, M. L. Littman, J. Huang, and D. L. Roberts, "A strategy-aware technique for learning behaviors from discrete human feedback," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-2014)*, 2014.

[9] A. Harutyunyan, T. Brys, P. Vrancx, and A. Nowé, "Shaping mario with human advice," in *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2015.

[10] M. Grześ and D. Kudenko, "Online learning of shaping rewards in reinforcement learning," *Neural Networks*, vol. 23, no. 4, pp. 541–550, 2010.

[11] S. Devlin, D. Kudenko, and M. Grześ, "An empirical study of potential-based reward shaping and advice in complex, multi-agent systems," *Advances in Complex Systems*, vol. 14, no. 02, pp. 251–278, 2011.

[12] K. Efthymiadis and D. Kudenko, "Using plan-based reward shaping to learn strategies in Starcraft: Broodwar," in *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pp. 1–8, IEEE, 2013.

[13] T. Brys, A. Harutyunyan, P. Vrancx, M. E. Taylor, D. Kudenko, and A. Nowé, "Multi-objectivization of reinforcement learning problems by reward shaping," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 2315–2322, IEEE, 2014.

[14] H. Kim, M. I. Jordan, S. Sastry, and A. Y. Ng, "Autonomous helicopter flight via reinforcement learning," in *Advances in neural information processing systems*, 2003.

[15] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*, pp. 363–372, Springer, 2006.

[16] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine learning*, vol. 22, no. 1-3, pp. 123–158, 1996.

[17] T. Brys, A. Nowé, D. Kudenko, and M. E. Taylor, "Combining multiple correlated reward and shaping signals by measuring confidence," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1687–1693, 2014.

[18] A. Harutyunyan, T. Brys, P. Vrancx, and A. Nowé, "Off-policy shaping ensembles in reinforcement learning," in *European Conference on Artificial Intelligence*, 2014.

[19] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[20] M. A. Wiering and H. van Hasselt, "Ensemble algorithms in reinforcement learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 930–936, 2008.

[21] H. R. Maei and R. S. Sutton, "Gq (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces," in *Proceedings of the Third Conference on Artificial General Intelligence*, vol. 1, pp. 91–96, 2010.

[22] M. Benda, V. Jagannathan, and R. Dodhiawala, "On optimal cooperation of knowledge sources - an empirical investigation," Tech. Rep. BCS–G2010–28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, WA, USA, July 1986.

[23] A. Harutyunyan, T. Brys, P. Vrancx, and A. Nowé, "Multi-scale reward shaping via an off-policy ensemble," in *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2015.

A COMPUTATIONAL MODEL OF GAIT CHANGES IN PARKINSON'S DISEASE PATIENTS PASSING THROUGH DOORWAYS

Vignesh Muralidharan, Pragathi Priyadharsini. B, V. Srinivasa Chakravarthy

Department of Biotechnology
Indian Institute of Technology, Madras
Chennai, India.
schakra@ee.iitm.ac.in

Balaraman Ravindran
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Chennai, India.

Simon J. G. Lewis
Brain and Mind Research Institute
The University of Sydney, NSW,
Australia

Ahmed A. Moustafa
Marcs Institute for Brain and Behaviour & School of Social Sciences and Psychology
University of Western Sydney
Australia

ABSTRACT

We present a novel Reinforcement Learning (RL) model of altered gait velocity patterns in Parkinson's Disease (PD) patients. PD gait is characterized by short shuffling steps, reduced walking speed, increased double support time and sometimes increased cadence. The most debilitating symptom of PD gait is the context dependent cessation in gait known as freezing of gait (FOG). Cowie et al (2010) and Almeida and Lebold (2010) investigated FOG as the changes in velocity profiles of PD gait, as patients walked through a doorway with variable width. The Cowie et al study reported a sharp dip in velocity, a short distance from the doorway that was greater for narrower doorways in PD freezers at ON and OFF dopaminergic medication. Almeida and Lebold also reported the same for ON medicated PD freezers and non-freezers. In this study, we sought to simulate these gait changes using a computational model of Basal Ganglia (BG) based on RL, coupled with a spinal rhythm mimicking central pattern generator model. In the model, a simulated agent was trained to learn a value profile over a corridor leading to the doorway by repeatedly attempting to pass through the doorway. Temporal difference error in value, associated with dopamine signal, was appropriately constrained in order to reflect the dopamine-deficient conditions of PD. Simulated gait under PD conditions exhibited a sharp dip in velocity close to the doorway, with PD OFF freezers showing the largest decrease in velocity compared to PD ON freezers and controls. Step length differences were also captured with PD freezers producing smaller steps than PD non-freezers and controls. This model is the first to explain the non-dopamine dependence for FOG, giving rise to several other possibilities for its aetiology. Analysing the influence of external factors on motor behaviour urges the need to understand gait at the level of the cortex, BG and the spinal cord. The study focuses on the contributions of the BG to gait impairment.

KEYWORDS: Gait, Freezing of Gait, Doorway, Basal Ganglia, Reinforcement Learning.

INTRODUCTION

Gait disorder is a common motor impairment observed in patients with Parkinson's disease (PD), a neurodegenerative disorder that involves a loss of dopaminergic neurons in the brain. It manifests in several ways including reduced stride length, reduced walking speed, increased cadence and increased double support duration [1]. One of the dramatic consequences of the pathology involves a paroxysmal inhibition in the ability to initiate locomotion, also known as freezing of gait (FOG). It is an episodic phenomenon associated with a sudden, intermittent cessation of locomotion which occurs in response to certain contextual situations. These may include situations such as a start hesitation (freezing while initiating a step), turn hesitation (freezing occurring while performing a turn) and destination hesitation (on reaching an endpoint). PD gait features such as reduced stride length and walking speed appear to be gradually aggravated under certain environmental conditions, culminating in a motor block, or a freezing episode [2]. In some cases freezing can be seen in contexts such as facing transverse lines on a road crossing or narrow confined spaces or doorways [3] while on the other hand the same transverse lines on a treadmill alleviates freezing symptoms [4]. This indicates that control over gait, influenced by higher level areas (cortical and subcortical) on the locomotor apparatus (spinal cord), needs to be properly understood.

An RL based model of the cortico-basal ganglia (BG) system that controls the gait rhythms is presented in this study. Two experimental studies [5, 6], that investigate the gait patterns of PD patients as they approach a doorway, are simulated using the model. The study of Cowie et al. (2010) shows a sharp dip in velocity as the PD patient approaches the doorway, a dip that becomes sharper in the case of narrower doorways [6]; this effect was more pronounced in PD patients (ON and OFF freezers) than in healthy controls. Almeida and Lebold (2010) consider a similar setup but compare the gait patterns of PD freezers with non-freezers in terms of step lengths [5]. The proposed BG model accounts for the above mentioned velocity profiles and gait features (stride / step lengths) of PD patients from these two experimental studies [7].

We model them at two stages of control: 1) the higher level of control representing the cortico-BG system, and 2) the spinal level central pattern generators (CPG) that translate the higher level gait commands such as velocity into gait rhythm (fig. 1b). The BG model is essentially simulated using the Actor-Critic architecture, with the difference that the Actor is modeled by the GEN (GO/EXPLORE/NOGO) policy [8-11]. The spinal CPGs are modeled by networks of adaptive Hopf oscillators [12], described elsewhere [7]. The model is used to simulate the results of two PD gait studies [5, 6]

THE MODEL

The environmental context of the experiments involves the healthy controls and the PD subjects to walk through a doorway of a specific width. The proposed model uses this as a base to make an agent (simulated subject) repeatedly approach a doorway, walking along a short path and compute the velocity profile along the length of the track. The agent aims at passing through the doorway without bumping into the sides of the doorway. The well-known tradeoff between accuracy and speed in motor function [13-15] might cause rapid approaches to the doorway, thus resulting in a collision. Therefore, in our model, the agent learns to reduce its speed in the vicinity of the doorway, which it does using RL mechanisms. The overall model architecture is as shown in Fig. 1.

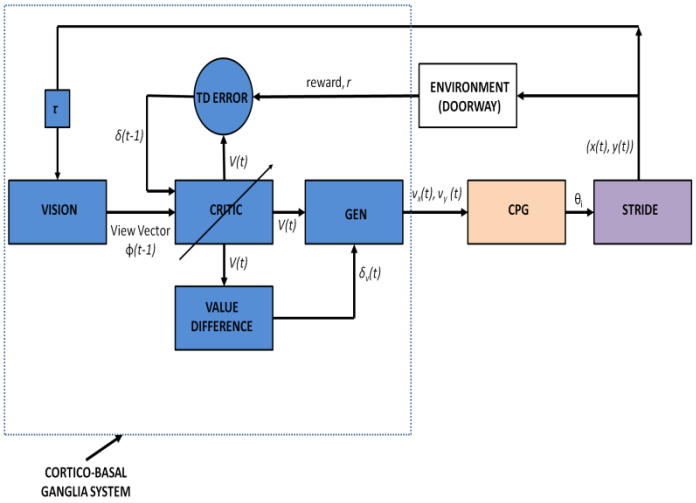


Fig 1 The model architecture with the cortico-BG system along with the CPGs

The model mainly consists of three components – the Cortico-BG system, CPG, and the locomotor apparatus. The Cortico-BG system, shown inside the dashed box (fig. 1), takes a representation of the view of the doorway, the 'view vector', from the position, *X*, of the agent. It is obtained from the cortical module: VISION. It is a 1x50 array of zeros and ones, where the position of ones denotes the part of the visual field where the doorway is situated. The block denoting τ denotes the time delay in the passage. The BG (consisting of the CRITIC, ACTOR (GEN), VALUE DIFFERENCE, and the TD ERROR modules) uses the view vector and updates the agent's velocity (v_x *and* v_y).

The Critic computes the value '*V*' for the view vector (ϕ(t),see [7] for details of view vector computation) as follows.

$$V(t) = \tanh(\sum W_i(t)\phi_i(t)) \tag{1}$$

The update equation for the above approximation (having weight vector, *W*) is given by

$$\Delta W = \eta\delta\phi(t) \tag{2}$$

Here, 'δ(t)' denotes the TEMPORAL DIFFERENCE (TD) error in value function, that is correlated to dopamine signaling [16]. The TD error with the discount factor γ is represented as follows.

$$\delta = r(t) + \gamma V(t) - V(t-1) \tag{3}$$

The policy (Actor) used in this study is the GO/EXPLORE/NOGO (GEN) policy. It represents an approach to action selection that involves stochastic hill-climbing over the value function space. Since the value peaks at the doorway in this scenario (reward is obtained at the doorway), the GEN can be used to climb the value gradient without bumping on the sides of the doorway. This form of actor in the BG differs from the classical RL implementation of Actor, wherein the action is typically modeled as an explicit function of the state ϕ, but the GEN policy computes the action by following the value gradient (value difference $\delta_V = V(t) - V(t-1)$) over the position space, *X*. Although value is a function of the view vector, ϕ(t), we perform the hill-climbing over the position space, *X*, that is mapped onto the view vector uniquely. The 3 discrete regimes – GEN [7] are modeled as below.

$$\begin{aligned}\Delta X(t) = {} & A_G\, sig(\lambda_G \delta_V)\, \Delta X(t-1) \\ & + A_E \chi \exp(-\delta_V{}^2 / \sigma_E^2) \\ & - A_N\, sig(\lambda_N \delta_V)\, \Delta X(t-1)\end{aligned} \tag{4}$$

where $sig(x_{sig}) = 1/(1+\exp(-x_{sig}))$, value difference term (δ_V) . The gain parameters of GEN include A_G, A_N, A_E for the GO, EXPLORE and NOGO regimes, respectively. The slopes of the sigmoids in the GO and the NOGO regimes are λ_G, λ_N, and the extent of exploration in the EXPLORE regime is controlled by σ. The first term in the eqn. (4) denotes the 'GO' regime that is significant when δ_V (VALUE DIFFERENCE) value is high. This means that the previous position update, X(t-1), had caused significant increase in the value and hence ΔX(t) is allowed to be in the same direction as ΔX(t-1). A low level of δ_V implies that the previous position update had caused significant decrease in the value. Therefore, the position update is in the opposite direction to the previous update, and thus a 'NOGO'. Whereas at the intermediate levels of δ_V, the policy randomly explores the position update (velocity - v_x *and* v_y) space (i.e., EXPLORE).

This velocity information coming from the GEN is sent to the CPG module which is a network of Hopf oscillators, translating this velocity into joint angles (θ_i). More information on the same can be found at [7, 12]. The subsequent block labeled STRIDE uses the θ_i and velocity information from GEN, to compute the next position. The ENVIROMENT (doorway) module checks if the new position results in a collision of the agent with the doorway. A positive reward, r = 5, is delivered if there is no collision, and a punishment (negative r = -1) in case of collision. The BG uses the view vector and reward information to compute value, thereby completing the cycle.

RESULTS

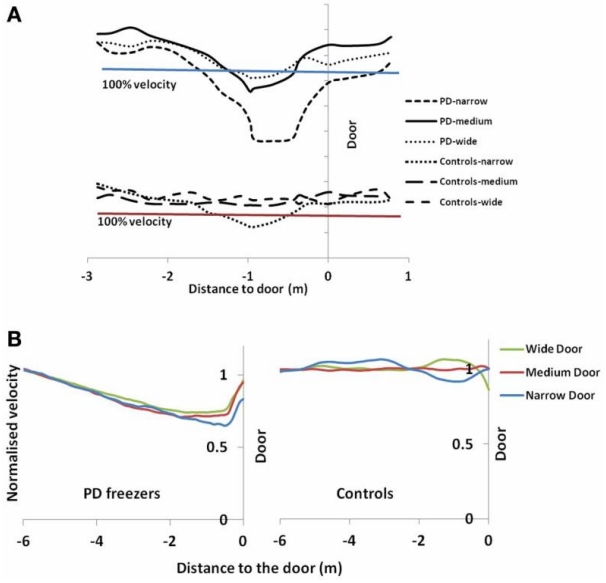


Fig 2: The normalised velocity profiles of the controls and PD freezers in the Cowie et al. experiment (A) and model (B). The 100% velocity in the experiment is the velocity exhibited under a no door condition and in the model the velocity is normalized using the average velocity far (5-6 m) from the doorway (Published in [7]).

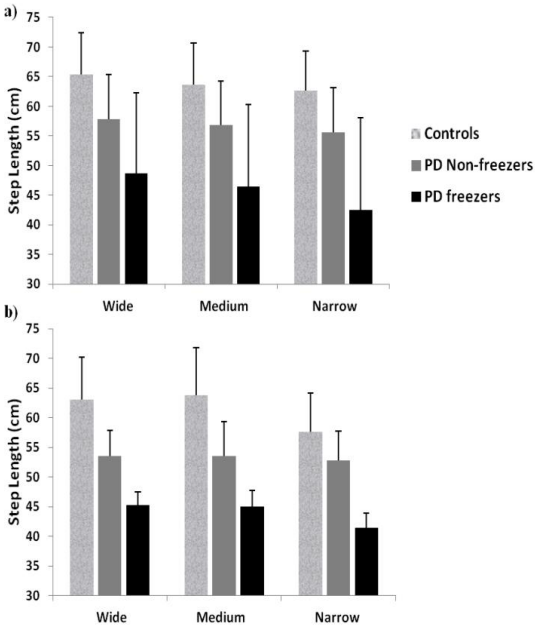


Fig 3: The step length profiles seen in the Almeida and Lebold experiment (A) and simulations (B) (Published in [7]).

The PD freezers show an exaggerated decrease in velocity on approaching a doorway, which seems to be amplified in the case of narrow doorways. The model is able to capture this behavior effectively. The velocity profile obtained from the model of Cowie et al. (2010) for controls and the PD freezers in the experiment and model is shown in Fig. 2a and 2b respectively. The velocity in the model has been normalised by using an average velocity far from the doorway. Additionally the Almeida and Lebold study

showed differences in step length between the controls, PD non-freezers and PD freezers. Our model captures this effect, where PD freezers show significantly reduced step lengths compared to non-freezers and controls as seen in Fig. 3.

DISCUSSION

Most models of BG take into account only the opponency of GO and NOGO signals arising from the direct and indirect pathways respectively [17, 18] to explain human behaviour. The current model necessitates the presence for an EXPLORE regime, needed for RL to take place effectively in the BG. In the model this regime is controlled by the parameter σ (extent of exploration) which is found necessary to explain the velocity dip seen before the doorway. It also seems to control the switch between the direct and the indirect pathways. The most interesting observation from the model is that the dopamine related parameters *alone* (simulated by clamping the temporal difference error in PD-OFF condition, and adding a dopamine medication constant in PD-ON condition) cannot explain the experimental behaviour. On exploring the roles of γ (discount factor earlier reported to represent the function of neuromodulator serotonin [19, 20]) and σ (exploration control parameter earlier reported to represent the function of neuromodulator norepinephrine [19]) in the velocity of gait, the model predicts the significant roles of non-dopamine correlates in FOG. The model suggests that in addition to clamping the TD error, the values of γ and σ had to be appropriately reduced in comparison to the controls for explaining the behaviour of PD freezers, whereas the PD non-freezers are differentiated from the freezers by having an increased σ. This is the first model to explain altered gait patterns using a cortico-BG system along with the spinal cord neural rhythm generators.

REFERENCES

1. Morris, M., et al., *Abnormalities in the stride length-cadence relation in parkinsonian gait.* Movement disorders, 1998. **13**(1): p. 61-69.
2. Chee, R., et al., *Gait freezing in Parkinson's disease and the stride length sequence effect interaction.* Brain, 2009. **132**(8): p. 2151-2160.
3. Rahman, S., et al., *The factors that induce or overcome freezing of gait in Parkinson's disease.* Behavioural neurology, 2008. **19**(3): p. 127-136.
4. Azulay, J.-P., et al., *Visual control of locomotion in Parkinson's disease.* Brain, 1999. **122**(1): p. 111-120.
5. Almeida, Q.J. and C.A. Lebold, *Freezing of gait in Parkinson's disease: a perceptual cause for a motor impairment?* Journal of Neurology, Neurosurgery & Psychiatry, 2010. **81**(5): p. 513-518.
6. Cowie, D., et al., *Insights into the neural control of locomotion from walking through doorways in Parkinson's disease.* Neuropsychologia, 2010. **48**(9): p. 2750-2757.
7. Muralidharan, V., et al., *A computational model of altered gait patterns in parkinson's disease patients negotiating narrow doorways.* Front Comput Neurosci, 2014. **7**: p. 190.
8. Chakravarthy, V.S., D. Joseph, and R.S. Bapi, *What do the basal ganglia do? A modeling perspective.* Biol Cybern, 2010. **103**(3): p. 237-53.
9. Kalva, S.K., et al., *On the neural substrates for exploratory dynamics in basal ganglia: a model.* Neural Netw, 2012. **32**: p. 65-73.
10. Magdoom, K.N., et al., *Modeling basal ganglia for understanding Parkinsonian reaching movements.* Neural Comput, 2011. **23**(2): p. 477-516.
11. Sridharan, D., P.S. Prashanth, and V.S. Chakravarthy, *The role of the basal ganglia in exploration in a neural model based on reinforcement learning.* Int J Neural Syst, 2006. **16**(2): p. 111-24.
12. Righetti, L. and A.J. Ijspeert. *Programmable central pattern generators: an application to biped locomotion control*. in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. 2006. IEEE.
13. MacKay, D.G., *The problems of flexibility, fluency, and speed–accuracy trade-off in skilled behavior.* Psychological Review, 1982. **89**(5): p. 483.
14. Bradshaw, E.J. and W. Sparrow, *The speed-accuracy trade-off in human gait control when running towards targets.* Journal of Applied Biomechanics, 2000. **16**(4): p. 331-341.
15. Duarte, M. and M.L. Latash, *Effects of postural task requirements on the speed–accuracy trade-off.* Experimental brain research, 2007. **180**(3): p. 457-467.
16. Schultz, W., *Dopamine signals for reward value and risk: basic and recent data.* Behav Brain Funct, 2010. **6**: p. 24.
17. Albin, R.L., A.B. Young, and J.B. Penney, *The functional anatomy of basal ganglia disorders.* Trends Neurosci, 1989. **12**(10): p. 366-75.
18. Frank, M.J., L.C. Seeberger, and C. O'Reilly R, *By carrot or by stick: cognitive reinforcement learning in parkinsonism.* Science, 2004. **306**(5703): p. 1940-3.
19. Doya, K., *Metalearning and neuromodulation.* Neural Networks, 2002. **15**(4): p. 495-506.
20. Tanaka, S.C., et al., *Serotonin differentially regulates short- and long-term prediction of rewards in the ventral and dorsal striatum.* PLoS One, 2007. **2**(12): p. e1333.

Combining Approximate Planning and Learning in a Cascade

Joseph Modayil, Kavosh Asadi, and Richard S. Sutton
Reinforcement Learning and Artifical Intelligence Laboratory
Department of Computer Science
University of Alberta

Abstract

A core competence of an intelligent agent is the ability to learn an approximate model of the world and then plan with it. Planning is computationally intensive, but arguably necessary for rapidly finding good behavior. It is also possible to find good behavior directly from experience, using model-free reinforcement-learning methods which, because they are computationally cheaper, can use a larger representation with more informative features. Our first result is to empirically demonstrate that model-free learning with a larger representation can perform better asymptotically than planning with a smaller representation. This motivates exploring agent architectures that combine planning (with a small representation) and learning (with a large representation) to get the benefits of both. In this paper we explore a combination in which planning proceeds oblivious to learning, and then learning, in parallel, *adds* to the approximate value function found by planning. We call this combination a *cascade*. We show empirically that our cascade obtains both benefits in the Mountain-Car and Puddle-World problems. We also prove formally that the cascade's asymptotic performance is equal to that of model-free learning under mild conditions in a prediction (policy evaluation) setting. Finally, another way in which learning may be advantaged over planning is that it can use eligibility traces. We show empirically that in this case the cascade is superior even if planning and learning share the same representation.

Keywords: Reinforcement Learning, Planning

1 Introduction

Planning is a mainstay of intelligent systems, both artificial and natural. In artificial systems, planning algorithms use computation and a model of the environment's dynamics to find good behavior, and planning as a kind of search has been essential to many AI successes. A^* search supports route-finding in navigation systems (Nilsson, 2009). Game-tree search has led to the best results in checkers (Schaeffer et al., 2007), poker (Bowling et al., 2015), and the game of Go (Gelly et al., 2012). Planning is also an essential aspect of natural intelligence. In this setting, planning refers to an animal's ability to rapidly modify its behavior using an approximate internal model of the environment, such as when an rat in a maze encounters an unexpected obstacle and rapidly decides on an alternative route to food (Tolman, 1948). Planning enables an agent to achieve good behavior through sheer computation when given an environmental model. If a model of the environment is not known a priori, it is sometimes possible to achieve good behavior by planning with an approximate model that is learned from experience (Abbeel et al., 2010). Learning a model requires additional computation, but it is arguably necessary to rapidly find good behavior when good models are not otherwise available. Methods that learn a model and plan with it to achieve good behavior are known as *model-based* reinforcement learning.

It is also possible in principle to learn good behavior directly from experience without using a model, through methods that are known as *model-free* reinforcement learning. Such learning methods are less sample efficient than using a model, as they adapt behavior only from immediate experience. However, they require much less computation, and so we can expect them to support a larger and more expressive representation. Through the use of a larger representation, they have the potential to obtain better performance asymptotically. This expectation motivates the exploration of an architecture that combines both planning and learning methods to obtain both benefits (rapid adaptation and high asymptotic performance).

Architectures that combine model-based and model-free learning have been studied both as abstract algorithms and as computational models for the behavior of brains. One successful algorithmic combination is known as Dyna (Sutton 1990), in which both model-based and model-free updates are made to the same value function in a tabular approach. This idea was extended to linear function approximation (Sutton et al., 2008, Parr et al., 2008), in which the model-based and model-free updates can be applied in parallel when they share the same representation and asymptotic solution. In psychology and neuroscience, researchers have explored novel ways of combining model-based and model-free methods: in competition (Daw et al., 2005), in weighted sums (Gläscher et al., 2010), and in co-operation (Gershman et al., 2014). These studies describe several ways to combine model-based and model-free methods, but they do not address the different computational costs of these methods under function approximation. Our approach is motivated by the need for a flexible way to combine the benefits of model-based and model-free methods under function approximation.

We first show through an example that model-based (planning) and model-free (learning) methods perform differently when using representations of different sizes and similar amounts of computation. Planning methods achieve good behavior rapidly, and learning methods achieve better behavior eventually. We propose a new architecture for combining learning and planning methods, in which the planning method constructs one value function with a small representation, and the learning method adds to it to make a better value function with a large representation. We call this architecture a cascade, and we demonstrate empirically that it has the merits of both methods. Further results show theoretical convergence guarantees and more benefits from learning with eligibility traces.

2 Problem formulation with linear function approximation

Reinforcement learning problems are often expressed with the formalism of a Markov decision process. A Markov decision process is written as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, T, d_0, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a finite set of actions, and $\mathcal{R} \subset \mathbb{R}$ is a set of real rewards. The environmental dynamics function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S} \to [0, 1]$ gives the probability of a transition from a state by an action, first receiving a reward, and then arriving at the next state. An agent starts at an initial state that is drawn from the distribution, $d_0 : \mathcal{S} \to [0, 1]$, and selects an action from each state according to some probabilistic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Using a policy for selecting actions at each state, the agent experiences a trajectory of states, actions, and rewards, $S_0, A_0, R_1, S_1, A_1, \ldots$, where $S_0 \sim d_0$. Problems are called episodic if the trajectories terminate after a finite number of steps, and this common case is handled within the same formalism by including an artificial terminal state from which all actions give a zero reward and return to the terminal state. The discount factor γ is used to define the return, $G_t \equiv \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i}$. The expected return, or value, is the focus of reinforcement learning algorithms. The value of a state under a fixed policy π is defined as the expected return from the state, $v_\pi(s) \equiv E_\pi[G_t | S_t = s]$. A similar quantity is defined for the expected return after taking an action from a state, namely $q_\pi(s, a) \equiv E_\pi[G_t | S_t = s, A_t = a]$, which is called the action value function of the policy π. Conversely, an (approximate) action value function can be used to define an ϵ-greedy control policy, namely the policy that at every state takes an action with the greatest value with probability $1 - \epsilon$, and otherwise (with probability ϵ) selects an action from the uniform distribution. We consider the control task of finding a policy that maximizes the expected return.

We are interested in the setting where the agent does not have the capacity to reason directly with the underlying states of the environment. This situation arises when the number of states is larger than the agent's memory, and also when the agent does not observe the state directly (also called a partially observable Markov decision process). In this setting, it is common to construct real-valued features from the information available to the agent at each state, and to approximate value functions as a linear function of the features, which is also known as linear function approximation. These features are represented with a vector, $\mathbf{x} \in \mathbb{R}^n$.

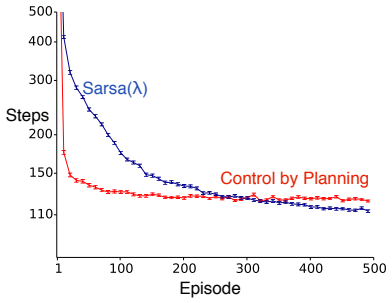
3 Solutions by learning and by planning

We first describe a model-free learning method, Sarsa(λ), for a control problem (Rummery and Niranjan, 1994; Sutton and Barto, 1998). The method uses state feature vectors given by $x : \mathcal{S} \to \mathbb{R}^n$, for which an element of the vector space is denoted $\mathbf{x}$. These features are used to approximate the action values for the agent's (current) policy π, $q_\pi(s,a) \approx \tilde{q}_L(x(s),a;\mathbf{w}_\bullet)$, where $\tilde{q}_L(\mathbf{x},a;\mathbf{w}_\bullet) \equiv \mathbf{w}_a^\top \mathbf{x}$, with a separate weight vector, $\mathbf{w}_a$, for each action. We use the notation of $\mathbf{w}_\bullet = \{\mathbf{w}_a\}_{a\in\mathcal{A}}$ for the set of all the weight vectors. For each experienced transition from $(\mathbf{x},A)$ to $(\mathbf{x}',A')$ with reward R, the learning method computes the temporal difference error, $\delta \leftarrow R + \gamma\tilde{q}_L(\mathbf{x}',A';\mathbf{w}_\bullet) - \tilde{q}_L(\mathbf{x},A;\mathbf{w}_\bullet)$, and changes the weights using this error. The quality and rate of learning are often improved through the use of eligibility traces, which are specified with a choice of $\lambda \in [0,1]$.

We now describe a model-based method for a control problem, namely an integrated approach to model-learning and planning from the reinforcement learning literature (Sutton et al., 2008). We define another feature vector mapping from states, $y : S \to \mathbb{R}^m$, for which an element of the vector space is denoted $\mathbf{y}$. We define a linear model of the expected next reward from a state by the vector $\mathbf{b}_a$, which approximates the expected next reward from state $S_t = s$ and action $A_t = a$ as $R_{t+1} \approx \mathbf{b}_a^\top y(s)$. We define a one-step linear model of the expected linear dynamics by the matrix $\mathbf{F}_a$, which approximates the expected next feature vector from state $S_t = s$ and action $A_t = a$ as $y(S_{t+1}) \approx \mathbf{F}_a y(s)$. Both the reward model and feature-vector transition model make predictions about the next time step and are updated using a one step learning rule. The value function $v(s)$ is modeled with a state-based approximate value function parameterized by a weight vector $\mathbf{v}$, namely $v(s) \approx \tilde{v}(y(s);\mathbf{v})$, where $\tilde{v}(\mathbf{y};\mathbf{v}) \equiv \mathbf{v}^\top\mathbf{y}$. Using these quantities, we define an approximate action value function, $\tilde{q}_P(\mathbf{y},a;\mathbf{F}_\bullet,\mathbf{b}_\bullet,\mathbf{v}) \equiv (\mathbf{b}_a^\top + \gamma\mathbf{v}^\top\mathbf{F}_a)\mathbf{y}$. The weight vector $\mathbf{v}$ is adapted from the reward and transition models by planning with the model, namely by simulating an expected feature vector transition that starts from the artificial feature vector, e_i, the ith unit basis vector.

Although we do not show the two methods in detail here, they require substantially different amounts of memory and computation at every time step. With respect to the number of features, the model-free method, Sarsa(λ), requires linear memory and computation, and the model-based method requires quadratic memory and computation for the matrix operations. Many factors can influence run-times of these methods, but under similar resource restraints, a learning method can support a substantially larger representation than a planning method.

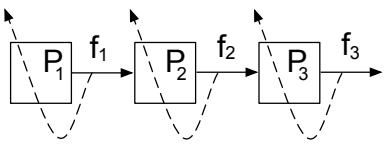
We tested both methods on the Mountain-Car domain, which is described by Sutton and Barto (1998). In this simulation domain, an under-powered car in a valley chose one of three actions (forward, reverse, coast) at each time step, with the objective of moving to the top of one hill as quickly as possible. The starting state was a zero velocity and a position near the bottom of the hill that was drawn from a uniform distribution over $[-0.7,-0.5]$. The discount factor γ was set to one and the reward was -1 per step, so the goal was to reach the terminal state in the fewest number of steps. The eligibility trace parameter λ was set to zero.



The methods were tested with representations of different sizes. The features were made by tile-coding (Sutton and Barto, 1998) with a 16×16 tiling of the state space. Two offset tilings were used for planning and ten offset tilings were used for learning, yielding binary feature vectors of dimension 512 and 2560 for planning and learning respectively. The different representations used by planning and learning led to both methods incurring comparable computational costs. In our implementation, the average time for one step of planning and learning was 12 milliseconds and 6.5 milliseconds respectively. As shown in the figure on the right, planning with the smaller representation rapidly achieved a good behavior, and learning with the larger representation achieved a better behavior eventually.

4 A cascade of planning and learning

Our desire to combine planning and learning methods with different representations led us to a simple and general architecture for combining adaptive processes. We define a *cascade* to be a series of ongoing processes, where each process is working towards a common goal, and each process is adapted locally from its output. The cascade is inspired in part by the earlier cascade correlation architecture for supervised learning (Fahlman and Lebiere, 1990). We show a general cascade as a block diagram on the right. Each process in the cascade is oblivious to the outputs of all the subsequent processes in the series, and is only impacted by the net output of previous processes. The adaptation in each process is local: the local output is used to update local parameters. In particular, each process is oblivious to the following processes.



We use the idea of a cascade to combine planning and learning methods, a cascade with just these two processes. As described previously, both methods generate an approximate value function with a linear function of the features. In this cascade, the planning method is first and adapts without knowledge of the subsequent learning method. The learning method improves on the action value function from planning by adding on to it, and the parameters are updated to reduce the error of the final value function.

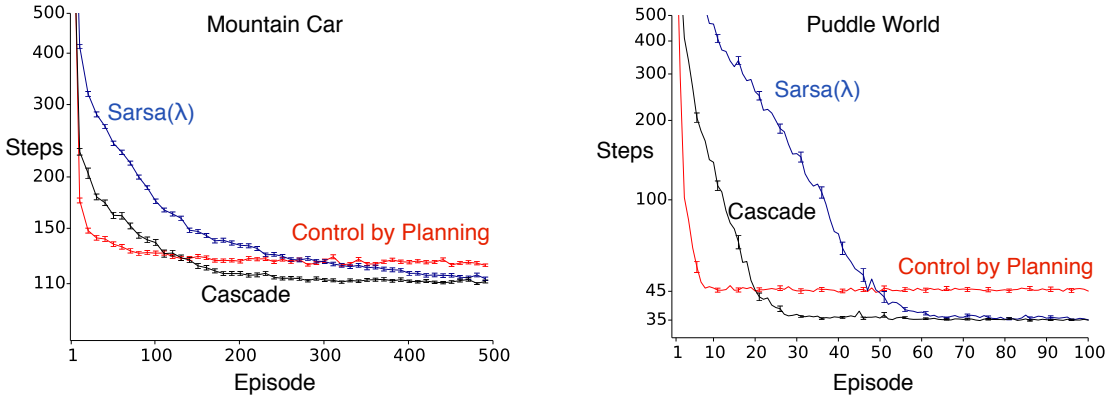


Figure 1: (Left) On the Mountain-Car problem, where the control objective is to minimize the number of steps, our proposed approach of the cascade has the merits of planning (Model-based Control) and learning (Sarsa(λ)). The graph shows that the cascade achieved a good initial behavior from planning with a small representation and the better asymptotic performance from learning with a larger representation. The error bars show the standard error of the mean computed over 300 independent runs. (Right) On the Puddle-World problem, the cascade achieved similar benefits. Again, a smaller representation was used for planning and a larger representation was used for learning.

More concretely, the planning method proceeds as usual and adapts its weights $\mathbf{v}, \mathbf{F}_\bullet, \mathbf{b}_\bullet$ to compute an approximate action value function $\tilde{q}_P$. The learning method builds on this function by constructing the cascaded action value function,

$$\begin{aligned}\tilde{q}_C(\mathbf{x}, \mathbf{y}, a; \mathbf{v}, \mathbf{F}_\bullet, \mathbf{b}_\bullet, \mathbf{w}_\bullet) &\equiv (\mathbf{b}_a^\top + \gamma \mathbf{v}^\top \mathbf{F}_a)\mathbf{y} + \mathbf{w}_a^\top \mathbf{x} \\ &= \tilde{q}_P(\mathbf{y}, a; \mathbf{v}, \mathbf{F}_\bullet, \mathbf{b}_\bullet) + \mathbf{w}_a^\top \mathbf{x},\end{aligned}$$

and only adapts the learning weights, $\mathbf{w}_\bullet$. The weights $\mathbf{w}_\bullet$ are adapted by using this action value to compute the temporal-difference error δ. Substituting in the approximate value functions for the cascade, we find that on an experienced transition from $(\mathbf{x}, \mathbf{y}, A)$ to $(\mathbf{x}', \mathbf{y}', A')$ with a reward of R, the temporal difference error δ is given by the following formula,

$$\delta \leftarrow R + \gamma \tilde{q}_C(\mathbf{x}', \mathbf{y}', A'; \mathbf{v}, \mathbf{F}_\bullet, \mathbf{b}_\bullet, \mathbf{w}_\bullet) - \tilde{q}_C(\mathbf{x}, \mathbf{y}, A; \mathbf{v}, \mathbf{F}_\bullet, \mathbf{b}_\bullet, \mathbf{w}_\bullet).$$

The temporal difference error is used to update the weight vector in learning by the same Sarsa(λ) algorithm, where the model-based contribution to the approximate action value function can be viewed as a bias for the model-free algorithm. Pseudocode for the cascade with this combination of model-based and model-free methods is shown on the right. The cascaded action value is used for ϵ-greedy action selection. Note that the model-based part (model-learning and planning) and model-free part (direct learning of action-values) operate as essentially decoupled processes, and the original planning and learning methods can be recovered from the cascade by fixing the weights of the other method to zero. The individual step sizes are written as α_L for learning and α_P for planning.

Pseudocode for planning and learning in a cascade

Initialize: $\mathbf{v}$, $\mathbf{F}_\bullet$, $\mathbf{b}_\bullet$,$\mathbf{w}_\bullet$
Set eligibility traces $\mathbf{z}_a$ to zero, $\forall a \in \mathcal{A}$
Receive initial feature vectors $\mathbf{x}$, $\mathbf{y}$, and take action A.
while episode not terminated **do**
  Receive reward R, and feature vectors $\mathbf{x}'$ and $\mathbf{y}'$
  #Control
  $A^\star \leftarrow \arg\max_a (\mathbf{b}_a^\top + \gamma \mathbf{v}^\top \mathbf{F}_a)\mathbf{y}' + \mathbf{w}_a^\top \mathbf{x}'$
  $A' \leftarrow A^*$ with prob. $1-\epsilon$, else from $\mathcal{A}$ at random
  Take action A'
  #Model-free learning of action values
  $\delta \leftarrow R + \gamma \left((\mathbf{b}_{A'}^\top + \gamma \mathbf{v}^\top \mathbf{F}_{A'})\mathbf{y}' + \mathbf{w}_{A'}^\top \mathbf{x}' \right)$
    $- \left((\mathbf{b}_A^\top + \gamma \mathbf{v}^\top \mathbf{F}_A)\mathbf{y} + \mathbf{w}_A^\top \mathbf{x} \right)$
  $\mathbf{z}_A \leftarrow \mathbf{z}_A + \mathbf{x}$
  $\mathbf{w}_a \leftarrow \mathbf{w}_a + \alpha_L \delta \mathbf{z}_a \quad \forall a \in \mathcal{A}$
  $\mathbf{z}_a \leftarrow \gamma\lambda \mathbf{z}_a \quad \forall a \in \mathcal{A}$
  $\mathbf{x} \leftarrow \mathbf{x}'$
  #Learning of one-step models
  $\mathbf{b}_A \leftarrow \mathbf{b}_A + \alpha_P (R - \mathbf{b}_A^\top \mathbf{y})\mathbf{y}$
  $\mathbf{F}_A \leftarrow \mathbf{F}_A + \alpha_P (\mathbf{y}' - \mathbf{F}_A \mathbf{y})\mathbf{y}^\top$
  #Planning with the learned models
  for p planning steps **do**
    for each feature index i **do**
      $M_{i,a} \leftarrow (\mathbf{b}_a^\top + \gamma \mathbf{v}^\top \mathbf{F}_a) e_i \ \ \forall a \in \mathcal{A}$
      $\mathbf{v}'(i) \leftarrow (1-\epsilon)(\max_a M_{i,a}) + \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} M_{i,a}$
    end for
    $\mathbf{v} \leftarrow \mathbf{v}'$
  end for
  $\mathbf{y} \leftarrow \mathbf{y}'$
  $A \leftarrow A'$
end while

5 Empirical Results

We applied our proposed cascade method to the Mountain-Car problem described previously. The step size parameters and representations were kept the same. Due to the spacing of the offsets in the tile coding, the cascade has a larger representation than either of its components, but this representation is not more expressive than the one used in learning.

The results for the cascade on the Mountain-Car problem are shown on the left graph in Figure 1. The cascade had better performance than Sarsa(λ) in the initial stages and also achieved the same asymptotic performance, but the cascade reached this performance level

faster. No algorithm parameters were changed for the cascade, the step sizes were 0.001 for planning and 0.1 for learning, and the number of planning steps was $p = 1$. These same parameters were used for the planning and learning methods. The mean computation time per step for the cascade was 16.6 milliseconds, slightly less than the sum of times for the component methods.

We performed another experiment in the Puddle-World problem, which is another standard domain for testing reinforcement learning algorithms with function approximation (Sutton, 1996). The underlying state state space was the unit square in $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, and there were four actions for moving the agent in the four cardinal directions. Each move was of length 0.05 plus a Gaussian noise with mean 0 and standard deviation of 0.01. Moves that would leave the square were truncated to the edges. The agent started each episode at the co-ordinates $(0.25, 0.6)$ and ended at a terminal state. The discount factor γ was set to one. The puddles had an oval shape and were located as described in the original paper. Moving through the puddles was costly, as the reward was -1 for each transition plus -400 times the distance to the edge of the puddle when the agent was in a puddle. The representations used for planning and learning were again made by tile-coding and of different sizes, where planning used a single 8×8 tiling and learning used ten 8×8 tilings. The value used for λ was 0.9, the number of planning steps was $p = 5$, and the step-size α for both methods was 0.1. The results for Puddle-World and Mountain-Car were qualitatively similar, as shown in Figure 1.

6 Additional Results

We have found some additional properties of the cascade approach to combining planning and learning. We have shown in the policy evaluation setting, that when the individual planning and learning methods converge, the cascade converges as well. When the features used for planning are representable as linear functions of the features used for learning, then the limiting performance of the cascade will be the same as for learning alone, so the benefits of planning come with no asymptotic harm. In addition, there are advantages for the cascade even when they use the same representations due to possibility of using eligibility traces in the model-free learning process. Eligibility traces enable asymptotically better temporally extended predictions under function approximation, and the improved predictions give rise to better control policies in some environments.

References

Abbeel, P., Coates, A., and Ng, A. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*.

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149.

Daw, N.D., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. Nat. Neurosci. 8, 17041711.

Fahlman, S. E., and Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*.

Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., and Teytaud, O. (2012). The grand challenge of computer Go: Monte Carlo tree search and extensions. *Communications of the ACM*, 55(3).

Gershman, S. J., Markman, A. B., and Otto, A. R. (2014). Retrospective revaluation in sequential decision making: A tale of two systems. Journal of Experimental Psychology: General, 143(1), 182.

Gläscher, J., Daw, N., Dayan, P., and O'Doherty, J. P. (2010). States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. Neuron, 66(4), 585-595.

Nilsson, N. J. (2009). *The Quest for Artificial Intelligence*. Cambridge University Press.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M. L. (2008). An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 752–759.

Rummery, G. A., and Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. (2007). Checkers is solved. *Science*, 317(5844):1518–1522.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, 1038–1044.

Sutton, R. S., Szepesvári, C., Geramifard, A., and Bowling, M. H. (2008). Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI)*, 528–536.

Tolman, E. C. (1948). Cognitive maps in rats and men. Psychological review, 55(4), 189.

Recurrent Neural Network Modeling of Anterior Cingulate Cortex Function

Danesh Shahnazian
Department of Psychology
University of Victoria
dshahnaz@uvic.ca

Clay Holroyd
Department of Psychology
University of Victoria
Holroyd@uvic.ca

Abstract

Despite decades of effort a unified theory of anterior cingulate cortex (ACC) function has yet to be realized. In particular, two seemingly incompatible classes of theory have emphasized a role for ACC in carrying out functions related to reinforcement learning: performance monitoring theories suggest a critic-like function, and action selection theories suggest an actor-like function. To reconcile these views, we recently proposed that ACC is responsible for option selection and maintenance according to principles of hierarchical reinforcement learning [1,2]. This position holds that the ACC learns the value of tasks, selects tasks for execution based on the learned values, and applies sufficient control over task performance to ensure that the selected task is successfully completed. Nevertheless, although this theory accounts for a wide range of empirical observations including the behavioral sequelae of ACC damage [3], it does not address abundant ACC single unit data nor influential neuroimaging findings related to conflict and surprise. Here we address this issue by implementing the proposed task control mechanism in a recurrent neural network architecture. The model simulates ensemble activity of ACC neurons at an abstract level, as well as univariate signals associated with surprise, conflict, and error processing. These simulations constitute a first step toward reconciling the action selection and performance monitoring theories of ACC function.

Keywords: anterior cingulate cortex, hierarchical reinforcement learning

1 Introduction

Despite decades of intense research on anterior cingulate cortex (ACC) there is still no consensus about the function of this brain region. Most theories of ACC function can be classified into either of two categories related to principles of reinforcement learning: performance monitoring theories suggest that ACC plays the role of a critic, and action selection and valuation theories propose that the ACC is an actor [4, 5]. For example, a prominent theory based mainly on functional neuroimaging data suggests that the ACC is involved in monitoring for response conflict [6], and single-cell recording in primates emphasize a role in action selection [7]. However, many of these theories do not adequately account for the behavioral sequelae of ACC damage, indicating that these putative functions are not uniquely implemented by ACC [2].

In view of this observation, we have recently proposed a novel theory that holds that ACC supports the selection and execution of extended behaviors according to principles of hierarchical reinforcement learning (HRL) [2]. This *HRL-ACC* theory holds that the ACC is responsible for the selection and execution of options, which are temporally abstract behaviors that describe context-specific action policies. According to the theory, ACC selects options for execution based on their learned values and then applies sufficient control over task execution to ensure that the selected option is completed successfully, as illustrated in recent computational simulations [3].

Despite these successes, the original HRL-ACC theory did not directly address a wealth of ACC-related data (such as conflict and surprise signals seen in human neuroimaging data [6, 8, 9], and complex patterns of single-cell activity observed in non-human primate studies [7]) because, according to the theory, these signals are correlated with but not causally related to ACC function [2]. Here we extend the HRL-ACC theory to account for these phenomena by training simple recurrent neural networks to execute options as series of stimulus-action sequences [10]. We then examine the dynamics of the network hidden units to provide a qualitative account of ensemble cell activity in rodent ACC [11], and leverage the error signal used to train this network to account for conflict and surprise signals observed in human ACC [6,9].

2 Model

The model is implemented in a simple, discrete-time recurrent neural network with 4 layers (Figure 1) [12]. The input layer receives information about relevant stimuli and action consequences in the environment. The output layer predicts the network input on the subsequent time step. A hidden layer maps the input at each time step to the output of the network on that time step, in the presence of information preserved in a context layer. In turn, the context layer stores the activation levels of the hidden layer units for one time step.

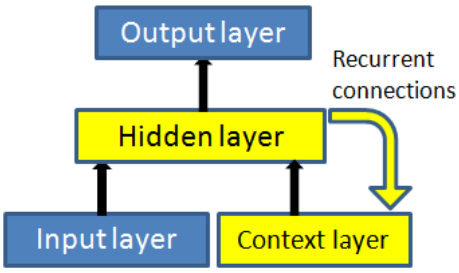


Figure 1. Network structure. Feedforward connections are depicted in black and recurrent connections are depicted in yellow [12].

Each unit is fully connected via excitatory projections to the units in its immediate downstream layer. The activation level of each unit is a logistic function of its summed inputs. The connection weights between units are initialized to random values between 1 and -1. For each simulation the network is trained by exposing it to repeated iterations of a set of target sequences using the back-propagation through time algorithm [13].

3 Simulation of ACC ensemble activity

ACC neurons are sensitive to a complex array of task-related events including stimuli, actions and rewards [14]. This neural activity appears to reflect dynamically evolving representations of task characteristics that are distributed across multiple cells (e.g., [11, 15]). Ma et al. [11] used a sequential lever pressing task to investigate ensemble coding of task-related information in ACC. Rats were required to perform three different sequences of three presses on three visually distinct levers to obtain reward (Figs. 2 and 3) [11]. Ma et al. found that lever presses of the same temporal order but of different spatial location are coded more closely in the state space than lever presses of the same temporal location but of different temporal order (Fig. 4). Because previous studies have shown that RNNs provide a useful means for analyzing such distributed representations in sequential tasks [12, 15, 16], we utilized a RNN to simulate the results of Ma et al. In particular, we trained the network to predict each step of each trial on the behavioral task, and then analyzed the activation levels of the hidden layer units as a proxy for distributed ACC activity. The network contained 50 hidden units, 50 context units, 7 input units (each corresponding to one sensory input: orienting in the direction of any of 3 levers, completion of lever press, and the sequence context) and 11 output units that predict what will occur on the subsequent time step (consisting of 9 combinations of 3 orientations and 3 stimuli types). Each trial was modeled as a sequence of events across 7 discrete time steps: the initial activation of a particular context unit (indicating which sequence to perform) as followed by 3 iterations of an orient-to-lever and pressing lever sequence, and ending with the activation of unit indicating reward delivery. The network achieved 100% accuracy after 6000 trials of training. To track the trajectory of the network through a task-related state space [12], we performed multi-dimensional scaling on the hidden unit activation levels during task execution and compared the results qualitatively to ensemble ACC neuron activity of rats executing the same task sequences.

We found that the different sequences produced trajectories through the network state space that were nearly parallel, a pattern that was consistent across runs that started from different initial conditions. Figure 5 shows the set of trajectories for a representative simulation. For both the empirical and simulated data, lever presses of the same temporal order in different sequences were coded in close proximity in the state space, whereas lever presses of the same spatial location but of different temporal orders were coded further apart (i.e., right lever presses in the 1st and 2nd sequence in Figure 4, vs. the 3rd lever press in the blue trajectory and1st lever press in the green trajectory in Figure 5). These results are consistent with the suggestion that ACC ensemble activity tracks the animal's progression through a "task space" [15].

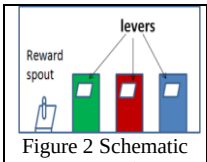


Figure 2 Schematic illustration of Ma et al. apparatus

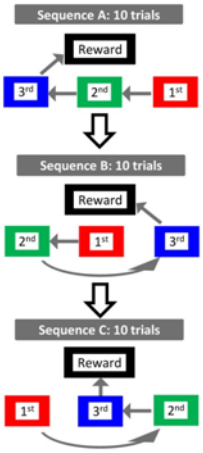


Figure 3 the spatial arrangement of stimuli relative to the levers [13]

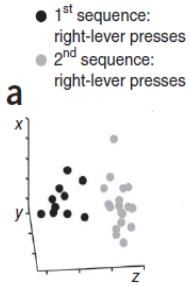


Figure 4 results of MDS performed on data recorded from ensemble of neurons in the ACC. Each dot represents the collective activity of 34 ACC neurons for a single subject. Levers of the same spatial location but of different temporal order are coded far apart [13]

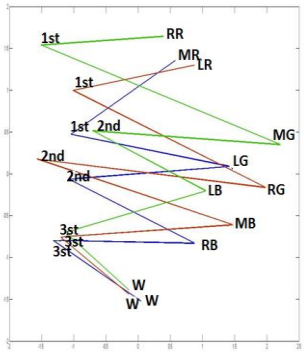


Figure 5 trajectory of hidden units' activation level revealed by MDS, each color indicates a different sequence. The numbers indicate temporal order within the sequence. The letters indicate the rest of network's output

4 Simulation of ACC univariate activity

A common finding in the human neuroimaging literature [6, 8, 17] is that ACC is activated by response conflict [6], errors [17] and otherwise unexpected events [17]. Of relevance to this issue, RNNs are predictive mechanisms: they are trained based on the current state to predict the immediately following state. Further, the error signal used to train the RNN reflects the discrepancy between these internal predictions and actual outcomes. Our conception of ACC as implementing option execution in an RNN therefore suggests that ACC implements an internal model that predicts the outcomes of actions at different stages of task execution, enabling the ACC to detect discrepancies between the

predictions of the internal model and external events. On this view, this *discrepancy signal* accounts for the surprise and conflict signals associated with the ACC BOLD response and with related event-related brain potential (ERP) components. In the following simulations, the discrepancy signal was calculated as the sum of the difference between the activation level of each output unit and its corresponding target.

We examined this issue by simulating the ERP response to novel stimuli (N2) and error commission (error-related negativity, ERN). Wessel and colleagues [8] utilized a paradigm that combined elements of a response compatibility task and an oddball task (Figure 6). On each trial subjects responded to an imperative stimulus indicating which of two buttons to press that was flanked by stimuli mapped either to the same or to a different response. The response was immediately followed by the appearance of a frequent "standard" stimulus on most trials, by an infrequent "novel" stimulus on a random subset of correct trials, or by a "target" stimulus on only 3 trials. As commonly observed, they found an enhanced ERN following error commission and large N2 to the unexpected novel stimuli (Figure 7).

To simulate these effects, we used a network comprised of 10 hidden units, 7 sensory units (representing the 4 possible flanker task stimuli and the standard, target and novel stimuli) and 4 output units (corresponding to predictions of the 2 possible responses and of the novel and standard stimuli).We exposed the network to a series of input-output mappings corresponding to the average participant performance on the task. For example, when presented with a flanker stimulus as input, the network predicted the agent's response as output. Note that each exposure emulated one trial in the actual paradigm and that the network was not trained on the task beforehand. Figure 8 shows the discrepancy signal to errors (ERN), standard stimuli, novel stimuli, and correct responses labeled as "baseline". The network produces relatively larger discrepancy signals to errors and to novel stimuli, which are both relatively unexpected, consistent with the empirical ERP data (Figure 7).

We also simulated the fMRI BOLD response to task events related to surprise, conflict, and errors in a "go-change" paradigm (Figure 9) [9]. These simulations follow the example of the predicted-response outcome (PRO) model of ACC function, which utilizes a dedicated prediction mechanism to account for such phenomena [9]. On each trial of this task the participants are first presented with a cue indicating the trial difficulty (low vs. high), followed by an arrow indicating which of two buttons should be pressed. On a fraction of these trials, called "change trials", the initial arrow is followed by a second arrow indicating that the participants should override their first response and press the other button; the remaining trials are called "go trials". The result of this experiment shows greater ACC activity on trials associated with high response conflict, error trials, difficult correct trials ("high error likelihood", HEL), and unexpected errors. Figure 10 shows these effects as simulated by the PRO model [17].

To account for these findings, we exposed the network to a sequence of input-output mapping corresponding to the average participant performance on the change-go task. For example, when presented with an arrow in the first time step, the network waits to see if a change signal is issued at the 2nd time step and then predicts the agent's response. The network was composed of 10 hidden units, 6 sensory inputs (corresponding to difficulty cues, two arrow directions, and two possible change signals), and two output units (corresponding to predictions of two possible responses). The discrepancy signal produced by the model qualitatively replicated the results reported in [17] (compare Figures 10 and 11).

The activity level of each output unit of a recurrent neural network reflects the probability that a particular event will occur at that time step, as determined by the frequency of occurrence of that event on that time step during the course of training [10]. For example, because most responses are correct, the model predicts correct responses; error responses are therefore unexpected and produce larger discrepancy signals. Likewise, on difficult trials associated with high error likelihood, correct responses are less frequent and therefore produce relatively larger discrepancy signals.

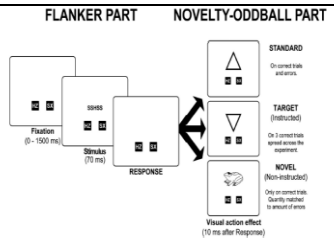


Figure 6 Wessel paradigm [8]

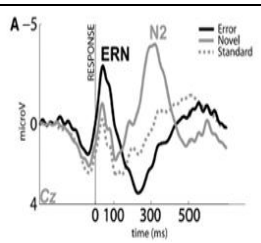


Figure 7 ERPs associated with error commission and novel stimuli [8]

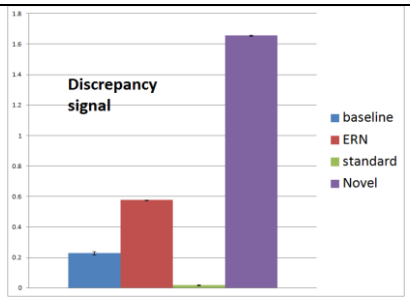


Figure 8 strength of discrepancy between network's output and actual event. Baseline is the average signal to correct responses

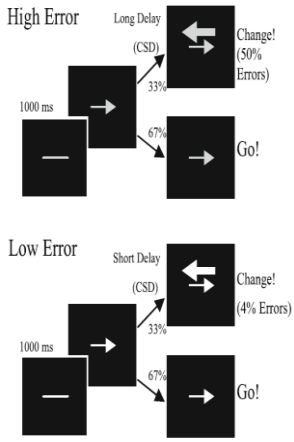


Figure 9 change-go paradigm [9]

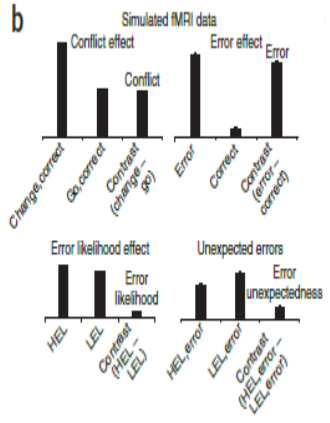


Figure 10 ACC's activity under different conditions [17]

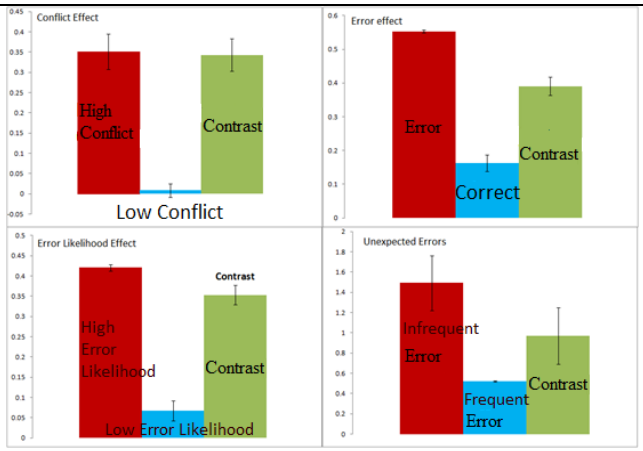


Figure 11 Strength of discrepancy between network's predictions and the actual event

5 Discussion

We propose that the ACC selects high-level options based on their learned reward values and then monitors the execution of those options by producing "surprise" signals when events deviate from the expected execution of the option. We previously instantiated the reward-learning component of the theory in an abstract model of rodent ACC function [3]. Here we extended the theory by implementing options in a RNN. These simulations provide a qualitative account of the ensemble activity of neural networks in ACC, consistent with previous observations that ACC tracks progress through subcomponents of task execution [11, 15]. The simulations also account for surprise, conflict, and error signals commonly observed in the ERP and the fMRI BOLD response that have been previously simulated with dedicated monitoring mechanisms [17]. These simulations constitute a first step in reconciling theories of ACC function based on actor/reinforcement learning vs. critic/performance monitoring.

Reference

[1] Holroyd, C. B., & Yeung, N. (2011). An integrative theory of anterior cingulate cortex function: Option selection in hierarchical reinforcement learning. The Neural Basis of Motivational and Cognitive Control, 333-349.
[2] Holroyd, C. B., & Yeung, N. (2012). Motivation of extended behaviors by anterior cingulate cortex. Trends in Cognitive Sciences, 16, 122-128.
[3] Holroyd, C. B., & McClure, S. M. (2015). Hierarchical control over effortful behavior by rodent medial frontal cortex: A computational model. Psychological Review, 122, 54-83.
[4] Holroyd, C. B., and Coles, M. G. H.. "Dorsal anterior cingulate cortex integrates reinforcement history to guide voluntary behavior." Cortex 44.5 (2008): 548-559.
[5] Silvetti, M., Alexander, W., Verguts, T., & Brown, J. W. (2013). From conflict management to reward-based decision making: actors and critics in primate medial frontal cortex. Neuroscience & Biobehavioral Reviews.
[6] Botvinick, M. M., Braver, T. S., Barch, D. M., Carter, C. S., & Cohen, J. D. (2001). Conflict monitoring and cognitive control. Psychological review, 108(3), 624.
[7] Kennerley, S. W., Walton, M. E., Behrens, T. E., Buckley, M. J., & Rushworth, M. F. (2006). Optimal decision making and the anterior cingulate cortex. Nature neuroscience, 9(7), 940-947.
[8] Wessel, J. R., Danielmeier, C., Morton, J. B., & Ullsperger, M. (2012). Surprise and error: common neuronal architecture for the processing of errors and novelty. The Journal of Neuroscience, 32(22), 7528-7537.
[9] Brown, J. W., & Braver, T. S. (2005). Learned predictions of error likelihood in the anterior cingulate cortex. Science, 307(5712), 1118-1121.
[10] Elman, J. L. (1990). Finding structure in time. Cognitive science, 14(2), 179-211.
[11] Ma, L., Hyman, J. M., Lindsay, A. J., Phillips, A. G., & Seamans, J. K. (2014). Differences in the emergent coding properties of cortical and striatal ensembles. Nature neuroscience, 17(8), 1100-1106.
[12] Botvinick, M., & Plaut, D. C. (2004). Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action. Psychological review, 111(2), 395.
[13] Williams, R. J., & Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. Back-propagation: Theory, architectures and applications, 433-486.
[14] Walton, M. E., & Mars, R. B. (2007). Probing human and monkey anterior cingulate cortex in variable environments. Cognitive, Affective, & Behavioral Neuroscience, 7(4), 413-422.
[15] Lapish, C. C., Durstewitz, D., Chandler, L. J., & Seamans, J. K. (2008). Successful choice behavior is associated with distinct and coherent network states in anterior cingulate cortex. Proceedings of the National Academy of Sciences, 105(33), 11963-11968.
[16] Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature 503(7474), 78-84.
[17] Alexander, W. H., & Brown, J. W. (2011). Medial prefrontal cortex as an action-outcome predictor. Nature neuroscience, 14(10), 1338-1344.

Human Reinforcement Learning in Non-Stationary Environments

Cameron D. Hassall
The Neuroeconomics Laboratory
University of Victoria
Victoria, BC
chassall@uvic.ca

Olave E. Krigolson
The Neuroeconomics Laboratory
University of Victoria
Victoria, BC
krigolso@uvic.ca

Abstract

Non-stationary environments are characterized by changes in the underlying reward structure. Detecting and responding to these changes can be challenging for reinforcement-learning (RL) systems, especially when feedback validity is low. Here we present the results of two experiments suggesting that RL in humans is dependent on knowledge about environmental uncertainty. Specifically, we had participants choose between two options with different reward probabilities. Reward probabilities were either static (a stationary environment) or occasionally reversed (a non-stationary environment). In contrast to previous work, our first experiment revealed no environment-dependent modulation of the feedback-related negativity (FRN), a component of the human event-related potential (ERP) thought to index an RL prediction error. However, when participants in a second experiment were cued as to which environment they were in, we observed the predicted enhancement of the FRN in non-stationary environments relative to stationary environments. These results suggest that while an RL system may be involved in uncertainty detection in humans, it's probably not the whole story.

Keywords: non-stationary environments, electroencephalography, feedback-related negativity, uncertainty

Acknowledgements

This research was funded by the Natural Sciences and Engineering Research Council of Canada.

1 Introduction

In order to maximize their utility, value-driven decision makers must first learn about their world, and then optimally exploit this information by selecting those actions that are most likely to lead to a reward. The world is an uncertain place though, and which choice is best may change. Reinforcement learning (RL) models are able to detect and adapt to shifts in reward payouts through the computation of a prediction error (a comparison between actual and expected outcomes) and the subsequent modification of the value associated with a choice. However, the lack of an explicit representation of uncertainty in RL models (i.e. model-free RL) limits their ability to rapidly detect shifts in reward probabilities, especially when feedback validity is low (that is, when the likelihood of a reward is low). Furthermore, others have pointed out that the predictions of model-free RL may not closely align with either human performance or neural data (Alexander & Brown, 2011). In other words, while there is evidence implicating a medial-frontal RL system in humans (Holroyd & Coles, 2002), the role of this system in detecting uncertainty is currently unclear.

Evidence for an RL system in humans comes from both electroencephalographic (EEG) and functional magnetic resonance imaging (fMRI) recordings made during feedback processing. For example, a negative deflection is observed in the human event-related potential (ERP)—averaged EEG in response to an event—following nonrewards, but not rewards (Miltner, Braun, & Coles, 1997). The voltage difference between rewards and nonrewards is called the feedback-related negativity (FRN), which is typically maximal at frontal-central scalp locations, and occurs 200–350 ms after feedback (see Walsh & Anderson, 2012, for a review). The FRN has been source-localized to anterior cingulate cortex (ACC: Miltner et al., 1997; Walsh & Anderson, 2012) and is thought to reflect an RL prediction error (Holroyd & Coles, 2002).

Given that the FRN indexes an RL prediction error, we might predict an enhancement of this component following an unexpected shift in reward probabilities (i.e. in non-stationary environments relative to stationary environments). Indeed, that is what Bland and Schaefer (2012) found when they varied uncertainty across blocks of a decision-making task. While this result might suggest that model-free RL is sufficient for detecting unexpected uncertainty, it's worth noting that Bland and Schaefer (2012) used feedback validities of either 73% or 83% (i.e. these were the likelihoods of winning if the optimal response was made). This choice of values was consistent with many previous studies on probabilistic reversal learning (e.g. 70%: O'Doherty, Critchley, Deichmann, & Dolan, 2003; 75% or 80%: Behrens, Woolrich, Walton, & Rushworth, 2007; 80%: Chase, Swainson, Durham, Benham, & Cools, 2010). In light of Bland and Schaefer's (2012) observation of an enhanced FRN in uncertain environments, we wondered if the same enhancement would be seen in the presence of reduced feedback validity (e.g. below 70%)—we hypothesized that it would not, since in such a scenario it would prove too difficult to distinguish between nonrewards indicating a reward probability shift, and nonrewards due to bad luck.

Our goal here was to investigate the neural mechanisms behind how humans deal with shifting reward probabilities (also called unexpected uncertainty). To do this we recorded electroencephalographic (EEG) and behavioural data in two decision-making experiments. In Experiment 1, participants were given no clue as to when reward probabilities might shift, while in Experiment 2 they were shown a cue indicating whether or not a probability shift might occur. Feedback validity was kept low at all times, and so we hypothesized that in Experiment 1 no difference would be seen between the FRN in stationary environments and the FRN in non-stationary environments. In contrast, we hypothesized that the cues provided in Experiment 2 would provide the top-down influence necessary to help the RL system sort out which nonrewards "counted" (i.e. which nonrewards indicated that the less optimal choice was being made), and as such Experiment 2 would yield the enhanced FRN in non-stationary environments as predicted by previous work.

2 Method

2.1 Experiment 1

We tested 20 university-aged participants, all of whom were volunteers who received extra course credit. All participants provided informed consent as approved by the Health Sciences Research Ethics Board at Dalhousie University. Participants were seated comfortably in front of a display and performed a simple gambling task. The task was a two-armed bandit (Sutton & Barto, 1998) in which participants were asked to maximize their reward. On each trial participants chose one of two slot machines, represented by squares of different colours. Participants were told that choosing one of the slot machines was more likely to result in a win compared to choosing the other slot machine. Unknown to participants, choosing the higher probability square resulted in a win with probability $P(\text{win}) = 0.6$, and choosing the lower probability square resulted in a win with $P(\text{win}) = 0.1$. Participants completed 24 blocks of 20 trials each. In half of the blocks, chosen at random, the colours of the squares would swap partway through the block. Within these blocks—called non-stationary blocks—the colour swap, or context shift, occurred randomly between trials 11 and 20. See Figure 1 for trial timing.

EEG data were filtered (0.1–25 Hz pass band) and rereferenced to the average of the two mastoid channels. Next, eye movements were corrected using the algorithm of Gratton, Coles, and Donchin (1983) and baseline corrected using the

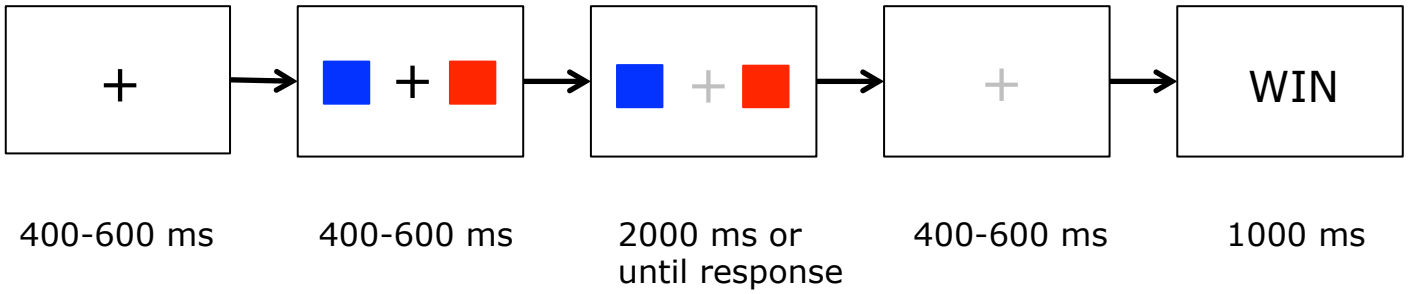


Figure 1: Experimental design, with timing details. Coloured squares could appear on either side. Choosing one coloured square was more likely to result in a win compared to selecting the other coloured square.

epoch 200 ms prior to feedback. Data were epoched around feedback events (200 ms before the event to 600 ms after the event) and any epochs with artifacts were discarded (i.e. epochs with a voltage change greater than 100 uV). Finally, we computed average epochs for each feedback type (wins/losses) both early and late within a block (trials 1–10 versus trials 11–20). Based on the grand average waveforms (e.g. Figure 3) and in line with previous research (Miltner et al., 1997; Holroyd & Coles, 2002) the FRN was defined as the mean voltage 200–400 ms post feedback, at electrode FCz. In addition to EEG, we recorded trial accuracy (whether or not the best bandit was chosen, regardless of outcome).

2.2 Experiment 2

Experiment 2 was carried out identically to Experiment 1 (using 20 different participants) with the exception that, prior to each block, participants were cued as to whether the block was stationary (no reward switch could occur) or non-stationary (a reward switch may occur). Block cues were images of two different casinos in which participants would be gambling—either an honest casino (stationary environment) or a dishonest casino (non-stationary environment). Participants were told that there was an honest casino and a dishonest casino, although they were not told which was which.

3 Results

3.1 Behavioural Data

Participants in both experiments improved over time (Figure 2), and eventually detected and responded to the reward reversal that occurred in the non-stationary environment. There was a slight performance advantage in the cued experiment relative to the uncued experiment (a significant main effect of experiment on mean performance: $F(1,38) = 19.13$, $p < .001$).

3.2 EEG Data

An FRN was observed in both Experiment 1 and Experiment 2. However, we noted an important difference between Experiment 1 and Experiment 2 with regards to how the FRN changed over time. In Experiment 1, there was no effect of environmental uncertainty on the FRN, ($F(1,19) = 0.241$, $p = .629$), and no time-environment interaction, ($F(1,19) = 1.011$, $p = .327$). In other words, the FRN responded the same way, regardless of environmental uncertainty. In Experiment 2, there was both a main effect of environmental uncertainty, ($F(1,19) = 6.145$, $p = .023$), and a significant time-environment interaction, ($F(1,19) = 7.489$, $p = .013$). See Figure 4 for a summary.

4 Discussion

The goal of these experiments was to investigate the degree to which an RL system in humans is engaged in detecting probabilistic reward reversals in highly uncertain environments, i.e. when feedback validity is low. Participants in both experiments learned optimal responses in a two-armed bandit task. However, participants in Experiment 2 were given the advantage of knowing whether or not the environment (block) they were in could include reversals. Besides

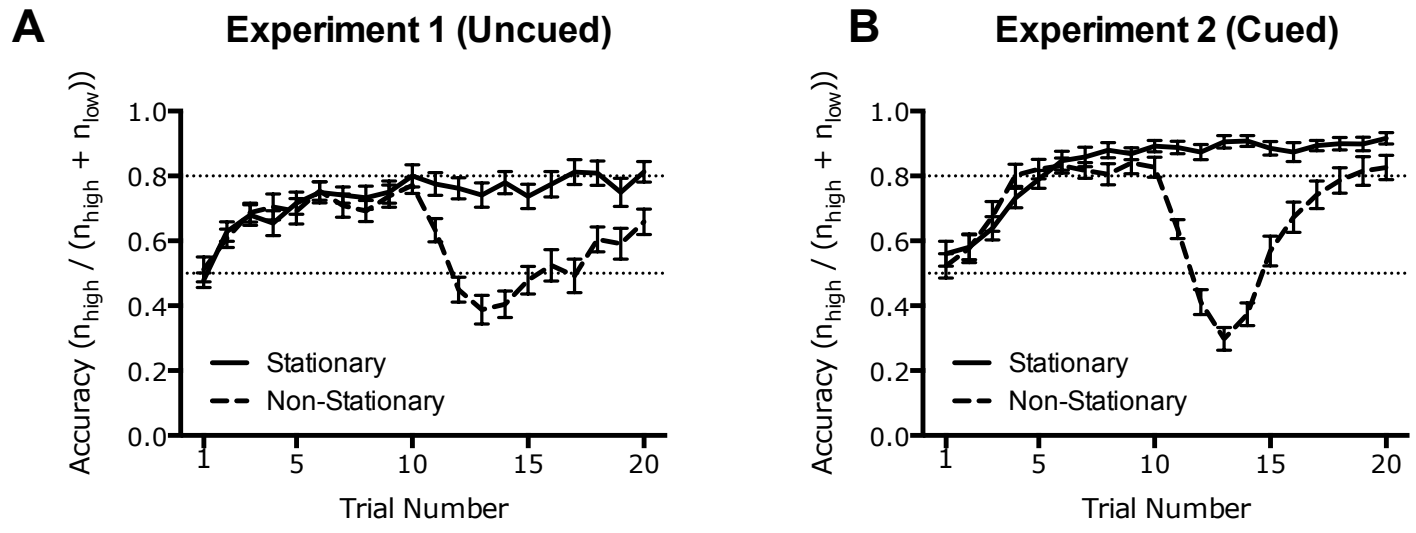


Figure 2: Mean accuracy, defined as the proportion of trials in which the optimal response was chosen, regardless of outcome. (a) In Experiment 1 participants were uncued as to which block type they were currently engaged. (b) In Experiment 2 block cues afforded a slight performance advantage relative to Experiment 1.

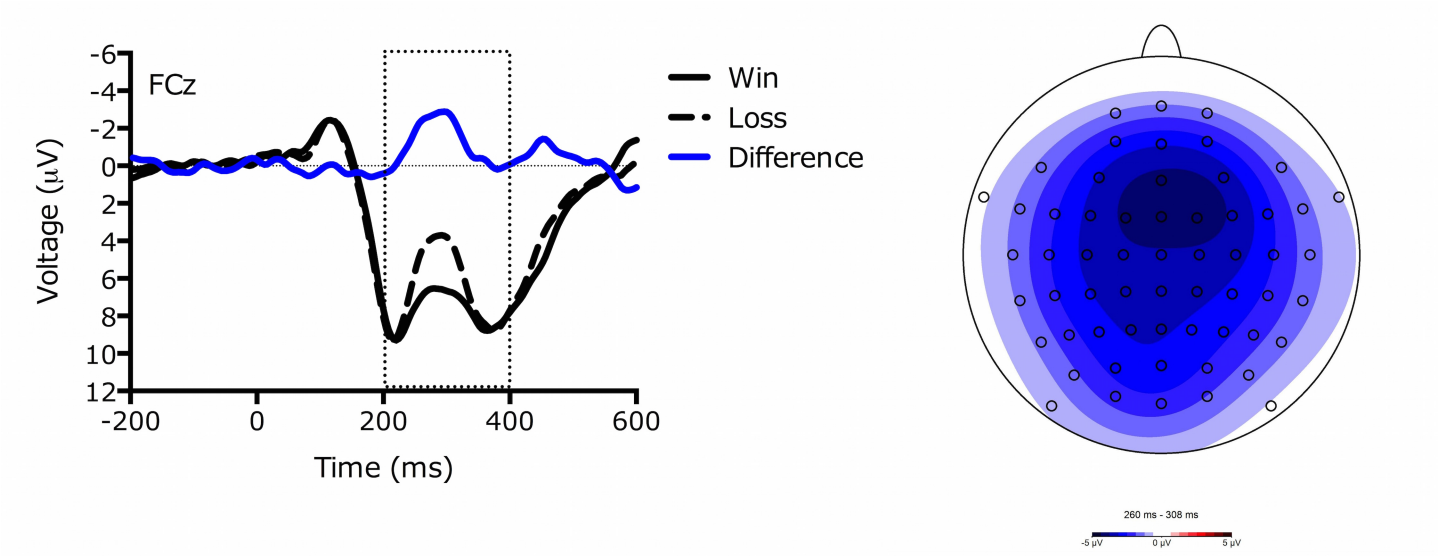


Figure 3: Sample ERP waveform response to feedback, with scalp topography. This is from Experiment 2 (Cued), non-stationary blocks only. Other conditions showed a similar timing and topography.

performing better, participants in Experiment 2 also showed a difference in the FRN, an ERP component thought to index an RL prediction error. In particular, the participants in Experiment 1 had the same FRN in each environment (stationary/non-stationary). In contrast, participants in Experiment 2 (who knew which environment they were in) showed an enhanced FRN in non-stationary environments.

While these results suggest the involvement of an RL system in humans when detecting reward reversals, they also suggest limits on such a system. In particular, our results suggest the need for an explicit representation of uncertainty in humans. Such a model-based system could, in theory, respond more rapidly to reward reversals than RL alone. For example, the detection of a reversal might signal the RL system to alter its parameters (e.g. learning rate) in such a way so as to expedite relearning. One influential model for uncertainty detection in humans is that of Yu and Dayan (2005). Their model simulates levels of two neurotransmitters: acetylcholine (ACh) and norepinephrine (NE). According to Yu and Dayan (2005), ACh levels signal feedback validity (also called expected uncertainty), and NE levels signal reward

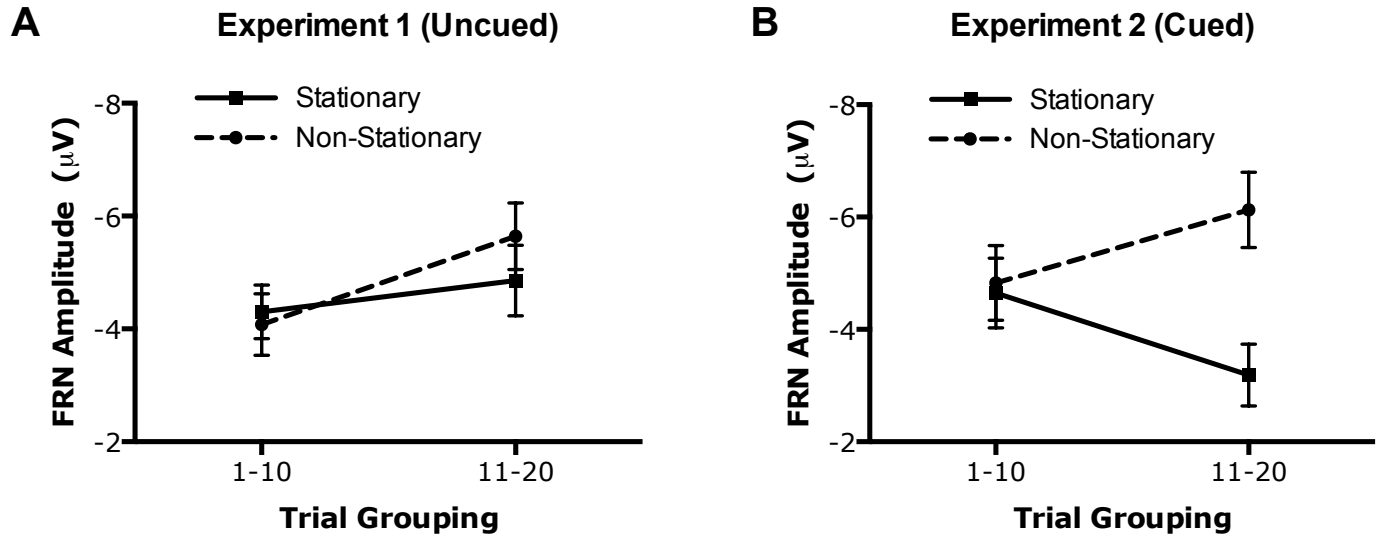


Figure 4: Summary of response to feedback. (a) When participants were uncued as to the uncertainty of the current block, there was no change in the FRN. (b) When cued as to block uncertainty, the FRN was enhanced for non-stationary blocks.

reversals (also called unexpected uncertainty). Thus, we might speculate about an RL system in humans (as indexed by the FRN) that works in conjunction with an ACh-NE system for uncertainty detection.

In summary, we provide evidence that uncertainty-based modulation of the FRN (a neural RL signal) is highly dependent on prior knowledge about the learning environment. Our results suggests that human RL is informed by an explicit system for uncertainty detection—a system that may be fooled when feedback validity is low, but performs better when you know whether or not you are in an uncertain environment.

References

Alexander, W. H., & Brown, J. W. (2011). Medial prefrontal cortex as an action-outcome predictor. *Nature Neuroscience, 14*(10), 1338-1344.

Behrens, T. E. J., Woolrich, M. W., Walton, M. E., & Rushworth, M. F. S. (2007). Learning the value of information in an uncertain world. *Nature Neuroscience, 10*(9), 1214-1221.

Bland, A. R., & Schaefer, A. (2012). Electrophysiological correlates of decision making under varying levels of uncertainty. *Brain Research, 1417*, 55-66.

Chase, H. W., Swainson, R., Durham, L., Benham, L., & Cools, R. (2010). Feedback-related negativity codes prediction error but not behavioral adjustment during probabilistic reversal learning. *Journal of Cognitive Neuroscience, 23*(4), 936-946.

Gratton, G., Coles, M. G., & Donchin, E. (1983). A new method for off-line removal of ocular artifact. *Electroencephalography and Clinical Neurophysiology, 55*(4), 468-484.

Holroyd, C. B., & Coles, M. G. (2002). The neural basis of human error processing: Reinforcement learning, dopamine, and the error-related negativity. *Psychological Review, 109*(4), 679-709.

Miltner, W. H. R., Braun, C. H., & Coles, M. G. H. (1997). Event-Related brain potentials following incorrect feedback in a time-estimation task: Evidence for a generic neural system for error detection. *Journal of Cognitive Neuroscience, 9*(6), 788-798.

O'Doherty, J., Critchley, H., Deichmann, R., & Dolan, R. J. (2003). Dissociating valence of outcome from behavioral control in human orbital and ventral prefrontal cortices. *The Journal of Neuroscience, 23*(21), 7931-7939.

Walsh, M. M., & Anderson, J. R. (2012). Learning from experience: Event-related potential correlates of reward processing, neural adaptation, and behavioral choice. *Neuroscience & Biobehavioral Reviews, 36*(8), 1870-1884.

Yu, A. J., & Dayan, P. (2005). Uncertainty, neuromodulation, and attention. *Neuron, 46*(4), 681-692.

Reinforcement Learning with Preferences

Johannes Feldmaier
Chair for Data Processing
Department of Electrical and Computer Engineering
Technische Universität München
johannes.feldmaier@tum.de

Hao Shen
Chair for Data Processing
Department of Electrical and Computer Engineering
Technische Universität München
hao.shen@tum.de

Dominik Meyer
Chair for Data Processing
Department of Electrical and Computer Engineering
Technische Universität München
dominik.meyer@tum.de

Klaus Diepold
Chair for Data Processing
Department of Electrical and Computer Engineering
Technische Universität München
klaus.diepold@tum.de

Abstract

In this work, we propose a framework of learning with preferences, which combines some neurophysiological findings, prospect theory, and the classic reinforcement learning mechanism. Specifically, we extend the state representation of reinforcement learning with a multi-dimensional preference model controlled by an external state. This external state is designed to be independent from the reinforcement learning process so that it can be controlled by an external process simulating the knowledge and experience of an agent while preserving all major properties of reinforcement learning. Finally, numerical experiments show that our proposed method is capable to learn different preferences in a manner sensitive to the agent's level of experience.

Keywords: Preference learning, prospect theory, affective state, reinforcement learning.

Acknowledgements

This work has been partly supported by IGSSE - International Graduate School of Science and Engineering, Technische Universität München, Germany.

1 Introduction

The expected utility hypothesis is commonly used to model human's decision making behavior in scenarios with uncertain outcomes, such as gambling. However, situations where preferences of individuals among the same choices are important, are not handled properly by the classic expected utility theory of Bernoulli. The Prospect Theory (PT), proposed by Kahneman and Tversky [7] introduces the concept of reference point to the expected utility theory of Bernoulli. This reference point enables to model preferences of individuals among same choices. It means that an internal reference point for a specific decision is essential to model the decision making behavior of people. Meanwhile, in psychology, a similar concept of affective states plays an important role in describing human's behavior. The human affect system is responsible to regulate the perception and assessment of events. It is able to assign rapidly emotions to occurring situations. This affective representation is then used to influence the decision making process, cf. [6].

Recently, it is considered that both affective and cognitive systems are essential in future smart systems, cf. [8]. The work in [2] also hypothesizes that rational decisions of humans are primarily influenced by so-called *somatic markers*, i.e. the positive or negative feeling towards a specific situation. Therefore, it is vitally important to consider both affective component and cognitive component, in order to design an autonomous and rational decision making agent. Because of these inseparable components involved in the human decision making system, we propose to integrate an externally controlled affective state into autonomous decision making. This results in a simulated emotional and rational experience of a reinforcement learning agent. There are examples (c.f. Section 3) where the level of experience influences the outcome of a task.

2 Related Works and Motivations

There are numerous works about the integration of affective and subjective components into Reinforcement Learning (RL) agents. We selected those, which have influenced our idea of integrating an internal affective state into RL.

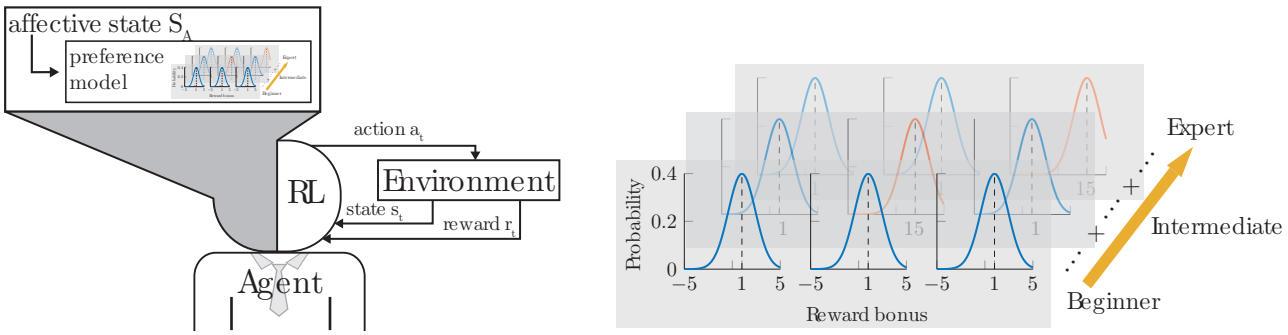
First of all, the work of Kenji Doya describes neuromodulatory systems and the (global) signals that regulate the (reinforcement) learning mechanisms of the human brain [3]. He argues that specific signals control and regulate the metaparameters (like randomness, action selection, reward prediction error, speed of memory update) for the reinforcement learning process. But there is no clear hypothesis regarding how the brain generates those signals. It seems to be a process separated of the actual learning running in different brain areas. This suggests that the brain has the capability of dynamically adjusting these metaparameters towards new or dynamically changing environments.

Besides these neurophysiological findings, the Prospect Theory shows the psychological influence of external and internal signals on the decision making behavior of human. In the Prospect Theory, a value function is described which is sensitive to deviations of the outcome (reward) according to a reference point in case of a risky choice. The reference point thereby is set by the current decision problem and depends rather on the losses and gains than on the final net asset value. This is also the reason why the framing of a choice problem becomes critical. The framing of the problem results in an external shift of the reference point which alters obviously the decision behavior [10].

A concrete combination of Reinforcement Learning and Prospect Theory is described by Ahn [1]. He has extended the conventional framework of RL for Markov decision processes (MDPs) with PT-based subjective value functions to model experienced-utility and predicted-utility functions. Furthermore, these functions vary dynamically according to the affective state of the decision maker (agent). This enables the agent to choose an action according to different risk attitudes and action tendencies on the basis of subjectively evaluated previous outcomes of decisions. The performance of his algorithm appears to be very good in the selected domains, but are difficult to reproduce due to the parameter dependence (which were additionally optimized for each domain). Moreover, the reference point of the PT value function is set automatically by the algorithm and an external control is not intended. Both, the external control as well as a strict separation of the reference point from the learning process seem to be important, if we consider the framing hypothesis and neurophysiological findings.

A third topic which has influenced our idea were preferences and biases. As preferences are fundamental for the human choice behavior, they are also an important component in learning. Human decision making is based on both, an objective and a subjective component [2]. Both components mainly depend on the emotional experience of a person. That means that decisions are evaluated in terms of objective and subjective rewards. The subjective rewards base on experienced feelings and emotions of previous outcomes of actions and decisions. Over time, specific situations and their past outcomes are associated with particular emotions (and their corresponding bodily changes). During decision making, these emotion-situation pairs are used as physiological signals (or *somatic markers*) to bias decision making towards certain policies while avoiding others. The whole set of *somatic markers* can be seen as the (emotional) experience of a human and is gathered during life. Regarding the development of artificial life-long learning agents such an emotional component is still a side issue. Therefore, the introduction of an experience-driven learning agent with specific preferences and aversions is vitally important to build more human like agents [12, 11].

A concrete approach combining RL with preferences is described by Fürnkranz et al. [5], where they combine preference learning [4] with RL. They learn a preference model from qualitative feedback and use it for ranking different policies



(a) (b)

Figure 1: (a) Reinforcement Learning framework with integrated preference model and affective state regulation. (b) Preference model using Gaussian distribution functions to add a reward bonus to a specific state or action according to the experience level S_A. In the depicted model there are three different experience levels. The actual preference in each level is highlighted (orange) and results in a reward bonus with a higher mean than the average. The different reward boni in each level are added up.

(fixed trajectory of a Markov process). The main drawback of this approach is the qualitative feedback which is used to evaluate already learned policies. This external qualitative feedback is given by human experts. Also, the learning process is directly interrelated with the preference model and there is no control how much the preference model affects the decision.

As we have seen the human learning process depends on external signals like framing effects and on internal signals generated by neurophysiological systems, preferences, and (emotional) experience. These are the influencing factors we want to use in our algorithm to model an agent with experience based preferences. Therefore, we propose a framework which should illuminate the role of an internal affective state (like an experience value) in the context of RL. We constructed this affective state so that it fulfills the Markov property, enabling the external control of the policy together with a preference model. In parallel, the affective state could also be used to model reference points as proposed in the prospect theory, enabling the agent to subjectively evaluate outcomes according to a (subjective) reference point. There are possible applications in the field of artificial intelligence, human-computer interfaces, and decision-support systems (recommender systems) where preferences of policies are needed and they are an essential aspect of rational autonomous agents.

3 Reinforcement Learning with Preferences

Basic Reinforcement Learning assumes a scenario in which an agent acts in a (finite) state space by performing different actions. A reward signal gives the agent feedback about its actions. The goal of RL is defined as maximizing the expected total sum of rewards. The basic formulation of a RL problem builds on the notation of a Markov decision process [9]:

- A set of states $S \in \{s_1, s_2, \ldots, s_i\}$ and a set of actions $A(s_i) \in \{a_1, a_2, \ldots, a_j\}$ which the agent can perform in a particular state s_i.
- Transition probabilities $\mathscr{P}^a_{ss'} = Pr\{s_{t+1}|s_t, a_t\}$ which denote the probabilities that an action a_t in state s_t leads to state s_{t+1}.
- A reward function $r(s_t, a_t)$ giving the agent reward according to the state and action performed.

We extended this basic RL framework with an additional state called affective state S_A (Figure 1a). According to this state the agent uses a preference model (Figure 1b) which gives a reward bonus to specific actions or states. In our experiment we use a one-dimensional state representation for controlling the preference model. For representing more complex affective states (e.g. general mood states) it would be possible to extend the state representation to a multi-dimensional vector controlling different preference models.

The basic idea underlying our preference model are various reward functions which are selected according to an affective state S_A. The additional reward functions correspond to reward facets which an agent can only perceive with increased experience or external feedback. The difference between our framework and approaches for multi-dimensional or dynamic reward environments surfaces in scenarios, where the agent first has to learn how the reward process works. For example, as a novice in cooking someone tells you to cook fried eggs and gives you eggs, a pan, and all other necessary equipment. You will start frying eggs until it looks like a fried egg. Now someone tastes and tells you that the yolk has to be cooked through. Up to now, as a beginner in cooking you only judged fried eggs according to the appearance, but now a new dimension is added: how the yolk has to be cooked. Next time cooking fried eggs (with an increased level of experience), this dimension is also considered and evaluated.

Another example, more complex especially for machines is the taste of coffee. As a beginner in drinking coffee you only judge your coffee according to the overall taste of bitterness or sweetness and probably the temperature. After drinking

coffee regularly you start to taste different flavors within a specific kind of coffee. Consequently, your experience in drinking coffee has added new dimensions of possible rewards to your preference model. As a coffee expert you choose your coffee according to a variety of different tastes and ways of preparation.

So, our preference model consists of several levels of reward functions which are only visible or accessible to the agent in a specific affective state (like a specific level of experience). We additionally introduce the constraint, that each level of experience can only increase, hence the additional reward functions are always added and can never be removed again (no-go-back-policy).

The signal generation representing the level of experience seems to be a substantial question. There are several solutions conceivable, like a monotonically increasing function according to the action taken by the agent or more advanced approaches considering the gathered knowledge and external feedback. Furthermore, according to neurophysiology a strict separation of the control signals for the learning process is desirable. That means, the internal affective signal (in the actual case the level of experience) must be externally controlled and should only be based partially on the results of the learned actions and outcomes. Ideally, it should base on past experiences, temporal effects, external feedback, and cognitive biases (like framing effects). Therefore, we decided to use in this stage of the development a simple stepwise function which increases the level of experience in three steps after a specific number of actions and does not interfere directly with the reinforcement learning process. This might be a oversimplification but allows a clear and concise formulation of the experiment.

4 Experiment

We conducted an experiment to investigate the properties of this extended reinforcement learning framework demonstrating that additional reward functions which are controlled externally can introduce preferences to the learned policy. In the experiment we simulated a three-armed bandit with Gaussian distributed rewards. At the beginning, each arm returns the same Gaussian distributed reward r_{ext} with $\mu_j = 1$ and $\sigma_j = 1$ for all arms $j = 1, 2, 3$. The experiment was repeated independently 100 times and 300 trials were played. Afterwards, the results where averaged. In this experiment, we used a simple affective state, which can be compared to a general level of experience. The state was generated externally and was modelled as an exponential increasing process simulating a continuously increasing level of experience.

We have used Q-learning with ϵ-greedy exploration [9] to learn the optimal policy in this experiment ($\epsilon = 0.05$, $\alpha = 0.8$, $\gamma = 0.4$). The results depicted in Figure 2 show that with increasing experience level the policy changes. At the first level (beginner, $0 \leq S_A \leq 0.5$), the agent is not able to differentiate the decisions and each action a_j is taken equally. After gaining some experience (intermediate, $0.5 < S_A \leq 0.95$) a second layer or dimension of reward functions is added to the learning process (second layer of Figure 1b). Now, the agent receives the original reward of the bandit process and an additional reward bonus (in the current setting also Gaussian distributed) which is added to the actual reward. In the intermediate level a preference model (or reward dimension) with $\mu_{2,int} = 15$, $\mu_{1,int} = \mu_{3,int} = 1$ and $\sigma_{1,int} = \sigma_{2,int} = \sigma_{3,int} = 1$ is added. So, a clear preference for action a_2 is introduced at this level of experience and the agent starts to prefer this action. In the expert level ($0.95 < S_A \leq 1$), a preference model is added which assigns a reward bonus, slightly higher than the one for action a_2, to action a_3 ($\mu_{3,exp} = 15$, $\mu_{1,exp} = \mu_{2,exp} = 1$ and $\sigma_{1,exp} = \sigma_{2,exp} = \sigma_{1,exp} = 1$). This results in a bias for action three which is therefore most frequently selected while the probability for action one and two decreases. The total reward r for updating the Q-function can be denoted as

$$r(s_t, a_t, S_A) = r_{ext} + \sum_{j \in A, k \in S_A} X_{j,k}, \qquad X_{j,k} \sim \mathcal{N}(\mu_{j,k}, \sigma^2_{j,k}), \tag{1}$$

where S_A denotes the set of experience levels, A the set of possible actions, and r_{ext} the external reward given by the three-armed bandit. $X_{j,k}$ is the reward bonus for a given action j in a specific affective state k which is in this example a sample of a normal distribution $\mathcal{N}(\mu_{j,k}, \sigma^2_{j,k})$.

5 Results

The results of the experiment as depicted in Figure 2 are straight forward and meet our expectations. But they show that the extension of the conventional RL framework with various reward process dimensions controlled by an external affective state can introduce preferences to the learned policy. In the introduction we motivated this extension by psychological and neurophysiological findings. This enables developers of learning agents to use it for developing agents with preferences, specific tastes and risk dispositions. The additional reward dimensions could be used e.g. for integrating prospect theory value functions besides conventional (like Gaussian) reward functions to simulate rational componentes of decision making (like risk aversion and attraction) while preserving the main underlying reward process maximizing the expected utility.

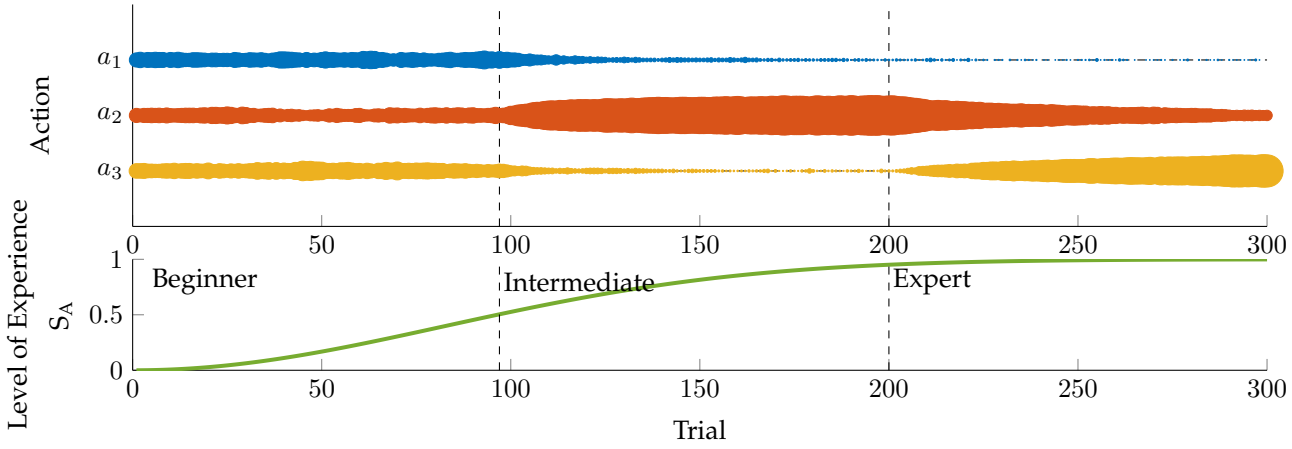


Figure 2: Learned policy for a simulated increasing level of experience. First, the agent acts like a beginner end selects every action equally (the thickness of the bars corresponds to the frequency of selection). At trial 98 the agent enters the intermediate state and can perceive an additional reward model with a preference for the second action (a_2). In the expert state, another reward model is added with a preference for action three (a_3).

6 Conclusion

The basic premise of the paper is that traditional RL can be extended by a preference model which is controlled by a single external state not interfering with the learning process. We also described in this paper the multi-dimensional approach of constructing a preference model with various reward functions as well as the application of it to integrate distinct preferences, biases, or cognitive frames into the framework of reinforcement learning.

In future studies we want to investigate the properties of such a framework regarding uncertainties in the reward process, the exploration and exploitation trade-off, and learning different "flavors or tastes" of polices. The property of an increasing reward function space would also be an interesting topic for further studies. Our vision is to build artificial agents with human-like preferences and sensitivity towards framing effects.

References

[1] Hyung-il Ahn and Rosalind W. Picard. Affective-cognitive learning and decision making: The role of emotions. In *The 18th European Meeting on Cybernetics and Systems Research (EMCSR)*, Vienna, Austria, 2006.

[2] António Damásio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Putnam Publishing, Kirkwood, NY, USA, 1994.

[3] Kenji Doya. Metalearning and neuromodulation. *Neural Networks*, 15(4):495–506, 2002.

[4] Jon Doyle. Prospects for preferences. *Computational Intelligence*, 20(2):111–136, 2004.

[5] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, 89(1–2):123–156, 2012.

[6] Gerd Gigerenzer, Ralph Hertwig, and Thorsten Pachur. *Fast and Frugal Heuristics - Theory, Tests, and Applications*. Oxford University Press, New York, NY, USA, 2011.

[7] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979.

[8] Marvin Lee Minsky. *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. Simon & Schuster, New York, NY, USA, 2006.

[9] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[10] Amos Tversky and Daniel Kahneman. The framing of decisions and the psychology of choice. *Science*, 211(4481):453–458, 1981.

[11] Juan D. Velásquez. When robots weep: Emotional memories and decision-making. In *Proceedings of the 15th National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, Madison, WI, USA, 1998.

[12] David Vernon, Giorgio Metta, and Giulio Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *Transactions on Evolutionary Computation*, 11(2):151–180, 2007.

Actively Learning to Attract Followers on Twitter

Nir Levine
Electrical Engineering
The Technion
Haifa, Israel
levinir@campus.technion.ac.il

Timothy A. Mann
Electrical Engineering
The Technion
Haifa, Israel
mann@ee.technion.ac.il

Shie Mannor
Electrical Engineering
The Technion
Haifa, Israel
shie@ee.technion.ac.il

Abstract

Twitter, a popular social network, presents great opportunities for on-line machine learning research. However, previous research has focused almost entirely on learning from passively collected data. We study the problem of learning to acquire followers through normative user behavior, as opposed to the mass following policies applied by many bots. We formalize the problem as a contextual bandit problem, in which we consider retweeting content to be the action chosen and each tweet (content) is accompanied by context. We design reward signals based on the change in followers. The result of our month long experiment with 60 agents suggests that (1) aggregating experience across agents can adversely impact prediction accuracy and (2) the Twitter community's response to different actions is non-stationary. Our findings suggest that actively learning on-line can provide deeper insights about how to attract followers than machine learning over passively collected data alone.

Keywords: Reinforcement Learning, On-line Learning, Contextual Bandits, Twitter

Acknowledgements

The research leading to these results has received funding from the European Research Council under the European Unions Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n.306638.

1 Introduction

Twitter is an on-line social network that allows people to post short messages (140 characters maximum), called status updates or tweets. It also allows users to read status updates posted by other users. Two additional actions are: (1) re-posting another user's status update, this action is called retweeting, and (2) favoriting a tweet, i.e. the tweet is marked as a favorite of the user. As one of the largest social networks, Twitter provides a great opportunity for evaluating machine learning algorithms on real-world data and evaluating them on-line. We target the problem of attracting followers in a community on Twitter and argue that actively learning can provide deeper insights than learning over passively collected data alone.

Without performing active experiments it is difficult to determine whether factors turned up in the analysis are only correlated, but not causally related, with attracting followers. Moreover these factors may change over time or vary depending on the history of an individual user. Therefore running experiments where algorithms control an account (rather than simply observing its behavior) can provide useful insights into the development of on-line relationships.

One difficulty of creating agents with the goal of acquiring followers is that naive exploitative strategies, such as mass following, are quite successful. This kind of aggressive following policy is easily labeled as bot behavior and is treated accordingly by the Twitter community. Our objective is to learn strategies that attract followers but avoid violating behavioral norms. By doing so, the agents receive followers based on providing a valuable service rather than exploiting other users.

Twitter has been the subject of intense research. Most of the work so far applied machine learning on passively collected data as opposed to learning from data collected on-line. However, collecting data on-line allows the agent to respond to the social network in real-time and choose actions most suitable to that time. This allows us to learn what causes users to follow (not just what is correlated). Although the problem of acquiring followers seems to be a popular subject on the Internet, we are unaware of any academic research that has examined an on-line approach for learning to attract followers.

The main contributions of this work are:

(1) We formalized the problem for learning to attract followers as a contextual bandit problem [3]. Our formulation encourages normative (rather than exploitative) behaviors, because the action space focuses on retweeting content (not following users).

(2) We executed a month long on-line experiment with 60 agents. Each agent interacted directly with the Twitter API and controlled a live Twitter account.

(3) We provide evidence for advantages of active learning over learning from passively collected data: the reward signal, based on the change in followers is non-stationary. Analyzing a data set collected in the past may result in poor performance, and more data is not always better. Although we had limited data for each individual agent, we found that aggregating data from multiple agents resulted in less accurate prediction. Therefore, learning should be applied to individual agents.

2 The Setting

One of the main contributions of this work is the development of a well-defined problem for actively learning in a social network. To our knowledge, no previous work has proposed a well-defined, active problem in social networks.

We allow a learning system to control a single user account on the Twitter social network. We will refer to this learning system as an agent. Our first intuition was to identify a multi-armed bandit problem [2], where the agent plays in a series of rounds. Let A be a set of $K \geq 2$ actions. At each round, the agent selects an action $a \in A$ associated with an initially unknown probability distribution over rewards. Once an action is selected, a reward is sampled from its corresponding distribution. The goal of the agent is to maximize its expected reward.

To formalize a problem as a multi-armed bandit, we needed to answer a few questions: (1) What are the rewards? (2) What are the actions available to the agent? (3) What is a round?

2.0.1 Rewards

Our objective is to design an agent that learns how to acquire followers. However, gaining followers is a rare event. A reward based entirely on the change in followers is difficult to learn from because the signal is sparse. Thus, we constructed reward signals from both changes in number of followers and other related events as explained in section 3.

2.0.2 Actions

The number of messages that can be expressed in 140 characters is too large to address directly. Additionally, users can perform other actions like following other users, retweeting existing status updates, and favoriting tweets, thereby significantly increasing the number of possible actions. However, a traditional multi-armed bandit problem assumes that the number of actions K is reasonably small.

In our experiment we used other users' content as actions rather than trying to invent new content. Furthermore, we restricted the set of tweets considered by the agent for retweeting to recent tweets containing the word "baseball". We chose the domain "baseball" because it has a large community but still represents a specialized domain where agents can learn to retweet valuable content.

Since in our setting it is very unlikely to sample more than once a specific action, we modeled the problem as a contextual bandit problem [3], which is a generalization of the multi-armed bandit problem. In each trial t, the algorithm observes a set $\mathcal{A}_t$ of possible actions sampled from a distribution ρ with support $\mathcal{A}$, where $\mathcal{A}$ is the collection of all tweets. Each action $a \in \mathcal{A}_t$ corresponds to a status update that can be retweeted and is described by a feature vector $x_{t,a}$. This feature vector $x_{t,a}$ describes properties of the action and the state of the environment. After choosing an action, the algorithm receives a reward r_{t,a_t} from which it learns and improves to the next trial. The objective of the agent is to learn a decision rule μ that maximizes

$$r = \mathbb{E}_{\mathcal{A}_t \sim \rho} \left[r_{t,a_t} \mid \mu \right] \ , \tag{1}$$

where r is the expected reward. In our experiment we use decision rules of the form

$$\mu(\mathcal{A}_t) = \arg\max_{a \in \mathcal{A}_t} f(x_{t,a}) \tag{2}$$

where f is a learned function that predicts the expected reward.

Due to the high-dimension of $X = \{x_{t,a}\}$, the feature space, we estimate rewards using function approximation/regression. The result of function approximation is a function $f : X \to \mathbb{R}$ that predicts the reward received for selecting the action $a \in A_t$ corresponding to $x_{t,a}$.

2.0.3 Rounds

A round could potentially be any constant length of time. However, enforcing normative behavior along with providing enough time between actions to allow other users to respond, led us to choose a time period of one hour between rounds.

A typical round in our setting is as following. First, the agent requests and receives a collection of recent tweets from the Twitter API about a specialized domain (e.g., baseball). Next, the agent examines the tweets it received from the Twitter API and retweets one of them. Finally, the agent sleeps for an hour and requests information about what changed. The agent then uses this information to calculate a reward and updates its decision rule. This finishes the round and a new round begins immediately.

2.0.4 The Exploration-Exploitation Dilemma

Exploration, trying actions with uncertain reward, is a critical issue in multi-armed bandit problems. We want the agent to choose an action that will have a high probability of attracting followers (exploitation), but the agent also needs to try various actions to learn what attracts followers (exploration). For simplicity, we use a simple the popular ϵ-greedy exploration strategy, which selects a random action with probability ϵ and the action with the highest predicted reward with probability $(1 - \epsilon)$.

3 Experiment

We designed and executed an experiment to determine whether simple machine learning algorithms could learn to attract more followers on Twitter than a random baseline. We created 60 Twitter accounts, each agent controlled one account. Every hour $t \geq 0$, each agent requested a collection of tweets $\mathcal{A}_t$ from Twitter. The agents selected a tweet $a_t \in \mathcal{A}_t$ to be retweeted (i.e., a status update) based on a list of features that were extracted from the tweets, x_{t,a_t}. One hour later a reward signal for a_t was computed by the agent. In line with our objective of maintaining normative behavior, agents only followed the user that posted a_t (before the agent retweeted it) with probability $P(follow) = 0.5$. The entire experiment was performed by using Twitter API.[1]

The reward signal used during the experiment was

$$r_{t,a_t} = \alpha_0 \Delta_{a,t} + \alpha_1 \Delta_{u,t} + \alpha_2 f_t + \alpha_3 w_t \ , \tag{3}$$

where $\Delta_{a,t}$ is the change in the number of agent's followers, $\Delta_{u,t}$ is the change in the number of followers for the tweet's original poster, f_t is the number of favorites the tweet received, and w_t is the number of retweets made to this tweet. The coefficients were $\alpha_0 = 100, \alpha_1 = 10, \alpha_2 = 10$, and $\alpha_3 = 1$, aligned with our objective we gave significantly higher weight to change in number of agent's followers.

Our experiment focused on tweets containing the string "baseball", because there is a constant flow of status updates and the topic is specialized enough so that an agent might learn useful knowledge about the domain. Status updates with offensive language were filtered before the agent made a selection. We divided the agents in three groups of 20: (1) uniform random (UR), (2) a gradient-based estimator (GE), and (3) a batch-based estimator (BE).

GE and BE estimated the reward signal for each status update from a collection of 87 features. We extracted features from the tweet and the user that posted it. We selected features based on prior work, such as [6, 5], along with others.

To encourage exploration, GE and BE selected a status update according to the uniform random rule with probability $\epsilon = 0.05$.

[1] `https://dev.twitter.com/docs/api/1.1`

3.1 Uniform Random (UR) Agents

The baseline UR algorithm selected retweets according to a uniform random distribution over the set of status updates. Thus, UR does not do any learning.

3.2 Gradient Estimator (GE) Agents

The GE algorithm incrementally updated a linear function approximator. GE applied gradient descent to minimize Mean Squared Error (MSE) and used a constant learning rate ($\eta = 0.1$). To better utilize the data, we introduced an adviser [4] to the GE algorithm. The adviser observes tweets not retweeted by the agent, and one hour later, at the same time the agent observes its reward, the adviser computes a modified reward signal

$$r'_{t,a_t} = \beta_0 \Delta_{a,u_t} + \beta_1 f_{a,t} + \beta_2 w_{a,t} \ , \tag{4}$$

for the other tweets where Δ_{a,u_t} is the change in the number of followers for the tweet's original poster, $f_{a,t}$ is number of favorites the tweet received, $w_{a,t}$ is the number of retweets made to this tweet, and coefficients $\beta_0 = 10, \beta_1 = 10, \beta_2 = 1$. Then, the adviser generates a hypothesis, using a batch training approach. The hypothesis generated by the adviser is weighted with the hypothesis generated by the agent. To reduce noise in the reward signal, GE only takes a learning step each 8 hours, averaging over all the hypotheses, to generate a new hypothesis as suggested by [1].

3.3 Batch-based Estimator (BE) Agents

The BE algorithm used Ordinary Least Squares (OLS) to train a linear function approximator after each round on all instances where the algorithm received the reward signal (3). All samples were weighted the same and the algorithm minimized MSE over the training samples. The BE algorithm used only samples it had selected to retweet (i.e., it did not use an advisor like the GE agents).

4 Experimental Results

We ran the experiment for about one month (May 9 – June 11, 2014) generating more than 600 status-updates for each account. At the end of the experiment, the average number of followers for the different groups was: 41.1 for UR, BE finished with 41.6, and GE had 44.95. Thus, GE acquired about 10% more followers on average than UR. A one sided T-test on the UR and GE groups shows that the difference is statistically significant with $p = 0.0291$ (i.e., 97.09% confidence that the two groups are generated by distributions with different means). However, the difference between UR and BE groups was not significant (see the next section for details). Two weeks before the end of the experiment, the average number of followers was 23.6, 23.6, and 23.8 for BE, UR, and GE respectively. However, over the last two weeks the difference between UR and GE grew 11 fold. This demonstrates that machine learning can attract more followers than a random strategy. It also raises the question: Why did GE outperform UR, while BE did not? In the next section, we analyze our experimental data to gain insight on attracting followers.

5 Importance of Active Learning

For comparison, throughout the analysis we normalized the rewards to have zero mean and a standard deviation of one. We noticed the poor performance of the BE algorithm. When we tested the performance offline, we found that Ordinary Least Squares (OLS) resulted in divergence. Therefore, while analyzing the data we used different function approximation models, Ridge Regression, LASSO, Elastic Net, and Support Vector Regression (SVR).[2] These methods apply types of regularization, thus gave better results (in off-line training).

In this section we show the importance of actively learning from data as opposed to learning from passively collected data. We identify two main problems in learning from passively collected: (1) the relationship between the features and reward function appears to be non-stationary, and (2) generalizing between agents tends to degrade prediction accuracy.

5.1 Nonstationarity

By examining the data, we found considerable evidence that the reward signal is a non-stationary function of our features. Thus training only on passively collected data is probably not satisfactory, because the best actions for acquiring followers seem to be time sensitive.

For each BE agent data (sorted chronologically) we trained and evaluated SVR model (achieved the smallest MSE). We divided the samples into chunks containing 100 sequential instances. The size was selected by experimentation yielding the smallest MSE. Each chunk was split into 75% training data and 25% testing data. Finally, we took the median MSE over all of the agents. The median MSE was 0.24, an improvement of 15% over when trained with all the data together. Thus, training with all data resulted in more error.

[2]The implementations used in our analysis are available at `http://scikit-learn.org`.

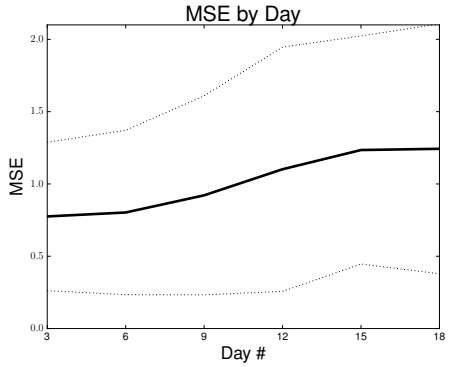


Figure 1: MSE of an SVR algorithm trained on the first 100 samples and tested on the remainder of the data. The MSE is plotted as a function of time (in days) after the 100 samples were collected. The error increases over time indicating that the reward signal we are trying to predict is nonstationary.

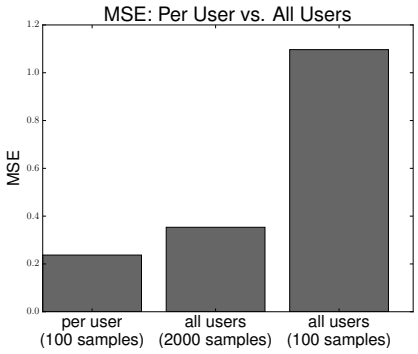


Figure 2: Comparison of MSE per user with 100 samples, all users with 2000 samples, and all users with 100 samples (75% training, 25% test). Combining data between users negatively impacts prediction accuracy.

Next we looked at predicting the number of followers rather than the reward signal (3) used in our experiments. For each agent, we trained SVR on the first 100 samples (sorted chronologically) and then used the remaining samples as test data. Figure 1 shows that the MSE increases as the experiment progresses. This is consistent with our hypothesis that the best strategy for attracting followers is changing over time.

These findings suggests that the reward signal is non-stationary, therefore learning from on-line data may result in more accurate predictions, as opposed to learning from passively collected data.

5.2 Generalizing Across Users

We examined the evolution of the weights learned by the GE agents. Specifically, we examined the median values and standard deviations for the weights between all the agents. The median values did not converge to a single point on weights space. On the contrary, the standard deviations increased over time, meaning the agents were learning different hypotheses.

Figure 2 shows the increase in error for BE agents when generalizing between agents compared to training on a single agent's data. We show the MSE for an SVR algorithm in three different cases, (1) per user with 100 samples, (2) combined data with 2000 samples, (3) combined data with 100 samples (using a moving window over the data). The MSE increases when generalizing over data from multiple users. When we used the same sample size (100) the MSE increased dramatically (around 4 fold). Even when we used a 20 times bigger training sets for the combined users' data (corresponding to 100 samples from each agent), the MSE still increased compared to the per user setup.

This demonstrates the importance of an agent learning from its own history. Simply generalizing over multiple agents' history actually degrades performance even with significantly more training data.

References

[1] Ofer Dekel and Yoram Singer. Data-driven online to batch conversions. In *NIPS*, volume 18, page 267, 2005.

[2] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[3] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA, 2010. ACM.

[4] Vinay N Papudesi and Manfred Huber. Learning from reinforcement and advice using composite reward functions. In *FLAIRS Conference*, pages 361–365, 2003.

[5] Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H Chi. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In *Social computing (socialcom), 2010 ieee second international conference on*, pages 177–184. IEEE, 2010.

[6] Dan Zarrella. Science of retweets. *Retrieved December*, 15:2009, 2009.

Expressing Tasks Robustly via Multiple Discount Factors

Ashley Edwards
College of Computing
Georgia Institute of Technology
Atlanta, GA, 30332
aedwards8@gatech.edu

Michael L. Littman
Computer Science Department
Brown University
Providence, RI 02912
mlittman@cs.brown.edu

Charles L. Isbell
College of Computing
Georgia Institute of Technology
Atlanta, GA, 30332
isbell@cc.gatech.edu

Abstract

Reward engineering is the problem of expressing a target task for an agent in the form of rewards for a Markov decision process. To be useful for learning, it is important that these encodings be robust to structural changes in the underlying domain; that is, the specification remain unchanged for any domain in some target class. We identify problems that are difficult to express robustly via the standard model of discounted rewards. In response, we examine the idea of decomposing a reward function into separate components, each with its own discount factor. We describe a method for finding robust parameters through the concept of *task* engineering, which additionally modifies the discount factors. We present a method for optimizing behavior in this setting and show that it could provide a more robust language than standard approaches.

Keywords: Reinforcement Learning, Reward Engineering, Generalization

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1148903.

1 Introduction

Software engineers are tasked with building programs that are robust to inputs. A sorting algorithm, for example, should work regardless of the size of the array it is operating on. Similarly, the *reward* engineering principle states that one should design rewards for reinforcement learning that are: general so the agent can operate in a dynamic environment, and allow the agent to be autonomous, without the need for interference from the designer [1]. If the size of the state space doubles, for example, one should not need to modify the rewards in order for the agent to still reach the goal.

Unfortunately, designing reliable reward functions is often difficult [2], particularly when there are multiple reward sources. For example, unintuitive reward values in the Maze problem generate drastically different policies [3]; an agent learning to ride a bicycle learned to ride in circles after receiving a positive reward for moving towards a goal [4]; and human generated rewards that were used for shaping caused "positive circuits" to form [5]. Such sub-optimal loops that distract one from reaching their goal have even been described in economic models of humans. We tend to over-indulge when there is an immediate reward and procrastinate when the reward is delayed (I'll quit smoking tomorrow; my diet can wait until after Thanksgiving, etc). If we were following the traditional definition of reinforcement learning, where the task is described as maximizing the total expected reward, then chain-smoking would be considered optimal behavior and so would over-eating. Thus, the reward engineer should have the freedom to determine which behaviors are truly desired, and rewards should rather be thought of as a bridge for producing these desired behaviors. Additionally, in order to be general, these reward functions should be designed well enough that they work across many perturbations of the environment. However, if poorly generated reward functions cause problems in static environments, it is unlikely that they will be able to generalize.

We have observed that rewards do not alone effect an agent's behavior, but rather the rewards coupled with the discount factor [5, 6, 7]. We believe that the difficulty is not necessarily that there are multiple rewards, but that the discount factors are entangled. It is impossible to discriminate the value of short term rewards from long term ones. We postulate then that we should also derive multiple discount factors when there are multiple rewards. A lower discount factor on short term goals could allow the agent to go to sub-goals, or to enjoy the benefits of shaping, without crippling its behavior.

We describe the approach of deriving robust rewards and discount factors as task engineering and make the following contributions: 1) we formally describe the notion of task engineering 2) describe how to derive parameters that allow the learning algorithm to perform well on a variety of instantiations of a task 3) derive an approach for using multiple discount factors with the learning algorithm Sarsa and 4) show problems where this approach is more robust than the standard Sarsa model with a single discount factor.

2 Preliminaries

A Markov Decision Process (MDP) is a tuple $\langle S, A, P, R \rangle$. The set S consists of the states in the environment, $s \in S$. The function A returns the actions $a \in A(s)$ that the agent can take in a state $s \in S$. The function $R(s, a)$ is the reward given for taking action a in state s, and the transition model $P(s, a, s')$ returns the probability of transitioning to state s' after taking action a in state s [8]. The goal of an MDP planning algorithm is to find a policy π that maps states to actions in such a way that it maximizes the long-term expected reward, that is, the value, of a state. The discount factor $0 \leq \gamma \leq 1$ encodes how rewards retain their value over time. An algorithm might seek to learn an action-value function which can be used for estimating the value of taking an action in a particular state. One such model-free algorithm is known as Sarsa, which makes the following updates to state-action pairs: $Q(s, a) = Q(s, a) + \alpha[R(s, a) + \gamma Q(s', a') - Q(s, a)]$.

3 The Problem of Multiple Rewards and Single Discounts

Reward functions in reinforcement learning are traditionally described as defining an agent's task [8]. This seems appropriate for monolithic reward functions, where a single reward is often used to motivate the agent to go to some goal [2]. However, when there are multiple rewards, it becomes less clear whether the reward function translates directly as the task. This distinction has often been seen in the literature, where metrics other than cumulative reward are used to determine whether a policy was successful or not. In particular, a learning algorithm might choose a policy that maximizes an intermediate reward rather than the reward at the goal. As such, rewards should be viewed as an intermediate representation for encouraging correct behavior, rather than the actual task. Thus, we follow terminology that distinguishes a task from the reward function [5]. We formally define what this concept means, and include the role of the task engineer.

3.1 Formalities

Define a *labeled MDP* as a set of states, actions, labels, transitions, and a labeling function. A labeling function maps states to labels. A *reward mapping* is a function that maps labels to scalar values. A set of labeled MDPs with the same label set is an *MDP family*. Define a *task* τ as a predicate on a history H of labels l. The *task objective* τ_o is completed when the predicate becomes true. The *behavior parameters* for a task, $\Theta^{(B)}$, are used to modify the agent's behavior, and include a reward mapping and discount factor. The *task parameters* of an MDP family F, $\Theta^{(F)}$, represent the physics of the MDP. Each label receives a *score* representing how desirable

it is for the task objective. The *task performance* of an agent for some task instantiation $\Theta_f^{(F)}$ for an optimal policy generated by some behavior parameter $\Theta_b^{(B)}$ can then be described as: $\tau_p(\Theta_b^{(B)}, \Theta_f^{(F)}) = \sum_{l \in H} score(l)$ That is, the task performance is the total overall score the agent receives during its policy. We can compute the *expected task performance* $\mathbb{E}[\tau_p]$ of $\Theta_b^{(B)}$ with respect to $\Theta_f^{(F)}$: $\mathbb{E}[\tau_p] = \frac{1}{|F|} \sum_{\theta_f^{(F)}} \tau_p(\Theta_b^{(B)}, \Theta_f^{(F)})$. Additionally, we define the *task efficiency* as how quickly the agent is able to solve a task for some task instantiation $\Theta_f^{(F)}$ for an optimal policy generated by some behavior parameter $\Theta_b^{(B)}$:$\tau_e(\Theta^{(B)}, \Theta^{(F)}) = \frac{1}{|H|}$. We can compute the *expected task efficiency* $\mathbb{E}[\tau_e]$ of $\Theta_b^{(B)}$ with respect to $\Theta_f^{(F)}$:$\mathbb{E}[\tau_e] = \frac{1}{|F|} \sum_{\theta_f^{(F)}} \tau_e(\Theta_b^{(B)}, \Theta_f^{(F)})$. The role of a *task engineer* is to find a $\Theta_b^{(B)}$ that maximizes both the expected task performance and the expected task efficiency. Later, we will describe an approach for finding these parameters.

3.2 On Robustness

Many types of MDP Families can be thought of as members of a broader class of MDPs, which we call $Hallway$ problems. In such tasks, the agent needs to get to a goal which is some amount of steps from the agent's location. The further away the goal is, the longer the hallway will be. The number of steps to the goal is directly effected by the task parameters, which might determine the number of obstacles in the domain, action dynamics, the initialization of the agent's location, or the location of the goal. In some types of hallways, intermediate rewards might be picked up along the way to the goal. In many cases, these rewards are used to encourage the agent's behavior. However, depending on how far away the goal is, they can often lead to distractions that hinder the agent from making progress. We have observed two types of hallways that might cause such distractions, which we will discuss below. In both of these hallways, the task objective is to go to the goal, which is located at the end of a hallway of length L. The behavior parameters that the task engineer can modify are the rewards and discount factors.

3.2.1 Positive Reward Hallways (PRHs)

In a Positive Reward Hallway (PRH), there are positive intermediate rewards, such as those used for sub-goals or reward shaping. Such hallways might introduce positive loop behavior where the agent becomes tempted to repeatedly receive some positive reward [4, 9, 5].

Suppose we have such a hallway where some state A allows the agent to repeatedly receive an intermediate reward of R_A and a terminal state B allows the agent to receive a reward of R_B. Suppose that R_B is some positive constant C times R_A, that is, $R_B = CR_A$. Then if the agent is currently located in state A, in order for the agent to choose going to B over staying in A, the following must be true: $\sum_{t=0}^{\infty} R_A \gamma^t < CR_A \gamma^{L-1}$. By rearranging and dividing out R_A, we get:

$$\frac{1}{1-\gamma} < C\gamma^{L-1}. \tag{1}$$

In order for this equation to be valid, i.e., for the agent to go to the goal in a large percentage of task instantiations we might need to greatly increase the value of C, especially as L increases. Even then, the scaling factor might not be large enough to motivate the agent. We cannot simply increase the value of γ to make the long-term value larger, because this will also increase the intermediate reward's value.

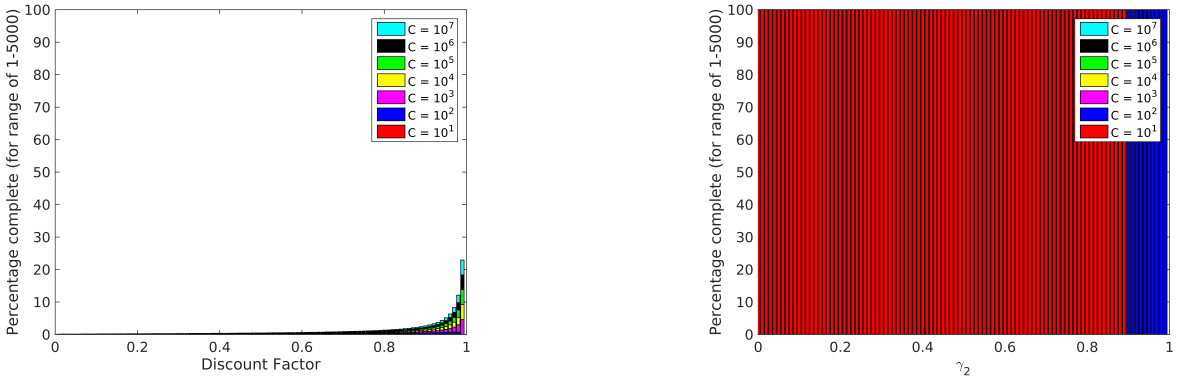
3.2.2 Negative Reward Hallways (NRHs)

In a Negative Reward Hallway (NRH), there are negative intermediate rewards, such as those used to inhibit undesirable behaviors. If there is some probability "slipping" and exhibiting such a behavior, then paralyzing loops may be introduced where the agent chooses to remain safe instead of risking taking a bad action.

Suppose we have such a hallway where some state P allows the agent to receive an intermediate reward of 0 and a terminal state Q allows the agent to receive a reward of R_Q. Suppose the agent receives a negative reward of R_U every time it reaches an undesirable state with label U. Assume that there is some expected value of U, $\mathbb{E}[U]$. This value is difficult to express because we do not know how often the agent will visit U. However, we do know that it is dependent on the probability of slipping. Additionally, the value of the discount factor directly effects $\mathbb{E}[U]$. If the agent is currently located in state P, in order for the agent to choose going to Q rather than staying in P, the following must be true:

$$0 < \mathbb{E}[U] + \gamma^{L-1} R_G. \tag{2}$$

In order for this equation to be valid, the right-hand-side of the equation needs to be positive. However, as the probability of taking undesirable actions increases, $\mathbb{E}[U]$ will become more dominant. This problem can be remedied by making the value of R_G very positive, but as the length of the hallway increases, the value of G decreases, and so the agent will become more likely to want to stay in state P. We might be tempted to decrease γ in order to decrease the expected value of U, but this approach would also decrease the goal state's value.



(a) Task performance for single discount factor

(b) Task performance for multiple discount factors

Figure 1: Here, the task objective is accomplished when the r.h.s. of the Equation 1 is greater than the l.h.s. We generated an MDP Family of task parameters whose lengths ranged from $1 - 5000$. We evaluated the Expected Task Performance for perturbed values of the discount factor and scalar C. For multiple discount factors 1b, we set γ_1 to 1 and perturbed $\gamma 2$.

3.2.3 Extending to Multiple Discount Factors

In both of these hallway problems, the discount factor causes conflicts for the desired policy. If γ is large, then the long-term value for states A and P will increase. If γ is small, then the value of state B will rapidly decrease. This becomes even more true as the length of the hallway L increases. This problem is clearly shown in Figure 1a. We do not have experiments for the NRH because the probability of hitting the wall is unknown, and so we cannot show how the discount factor will effect U's value. Still, intuitively, if we decrease U's value by too much, then the agent might exhibit more undesirable behaviors, and have poor task performance.

Suppose we use a distinct discount factor for each reward function, say γ_1 for the intermediate states and γ_2 for the goal states. Notice that now these state values will no longer be tied together. We can make $\gamma_1 < \gamma_2$, which will allow the intermediate state's value to be discounted more rapidly than the goal state's value. We can appropriately treat the intermediate reward as a short term reward without having any effect on the long-term goal's value. Additionally, if we make γ_2's value very large, then the goal's expected value will no longer be as dependent on L. The effect of this is clearly shown in Figure 1b.

These examples suggest that by using multiple discount factors we can more robustly solve task parameter instantiations where the behavior parameter instantiations might introduce distractions. By using multiple discount factors in such problems, we will be able to decrease a distracting state's long-term value without destroying the performance when the task parameters change.

4 Approach

We now show how to optimize behavior for such reward functions. We define a Multiple Reward Component MDP (MRC-MDP) as a tuple $\langle S, A, P, \mathbf{R} \rangle$, where S, A, and P are as before, but $\mathbf{R}$ is now defined as a *vector* of reward components, each with its own discount factor. This problem was solved in prior work [10, 11], however, both of these solutions require one to have the transition model, which may be infeasible. Therefore, we provide a method for using multiple discount factors with Sarsa.

We follow an approach where the Q-function is decomposed into N separate Q functions for each reward function [12]. We then provide a distinct discount factor for each Q-function. That is,

$$Q_i(s, a) = Q_i + \alpha[R_i(s, a) + \gamma_i Q_i(s', a') - Q_i(s, a)].$$

We can then take the overall sum of these components for a global Q function, which can be used for action selection. We call this approach Decomposed Sarsa with Multiple Discount Factors, or DECS-MDF.

Because we are interested in engineering agents whose maximized reward leads to correct task performance, the role of the task engineer is to find rewards and discount factors that produce this desired output. Even if we find behavior parameters that work for a particular setting, these parameters might not work for a wide range of variations of the task parameters. Therefore, we follow an approach that searches for parameters that do well in a sample of the possible members of the MDP family [13]. This work exhaustively searches for rewards that yield high performance. However, brute force is computationally expensive. We extend upon that work by using optimization and modifying the search parameters to also include multiple discount factors.

Searching for the parameters requires evaluating the parameters on their fitness, which is a value that is often used in optimization problems such as Hill-Climbing and Simulated Annealing [3]. Typically, one only needs to plug a fitness function and a method for choosing neighbors of the current parameters into these types of optimization problems. The steps for evaluating the fitness of

the parameters are as follows: 1) Randomly sample a set of task parameters F from the Hallway MDP family 2) Plug the current parameters $\Theta_b^{(B)}$ into the learning algorithm 3) Calculate the expected task performance $\mathbb{E}[\tau_p]$ as the current fitness.

5 Related Work

As we have seen, additional rewards may introduce locally optimal policies that distract the agent from completing the task [4, 5, 9]. Such issues have been addressed by using potential-based shaping functions that maintain the desired behavior for the optimal policy [9]. The idea of "positive loops" is similar to human's tendencies to over-indulge or under-indulge. Thus, economists often weight future and immediate rewards separately in economic models [14, 15]. Many works have shown that the choice of the discount factor effects the agent's policy [5, 7]. It has even been shown that poorly chosen discount factors might be the cause of loops in human generated reward [5]. Other works have addressed solving MDPs with multiple rewards and discount factors [10, 11]. However, these approaches require a transition model.

There are some works that evaluate the robustness of reinforcement learning algorithms. Pareto fronts have been used for evaluating multi-objective functions [16]. In another approach, reward functions that receive high fitness in a variety of MDPSs are exhaustively searched for, where each state in a history of trials receives a score for reaching some criteria [13]. Our approach for optimizing behavior parameters expanded upon this work.

6 Conclusion

We have shown that multiple discount factors might provide a robust language for solving tasks where multiple rewards might act as a bridge for guiding the agent to the goal. We provided a method for solving such problems, and showed why it could potentially be robust to perturbations of the task. We hope to extend these ideas to larger efforts in reward engineering.

References

[1] D. Dewey, "Reinforcement learning and the reward engineering principle," in *2014 AAAI Spring Symposium Series*, 2014.

[2] M. J. Mataric, "Reward functions for accelerated learning," in *In Proceedings of the Eleventh International Conference on Machine Learning*, pp. 181–189, Morgan Kaufmann, 1994.

[3] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, vol. 2. Prentice Hall Upper Saddle River, New Jersey, 2003.

[4] J. Randløv and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping," in *Machine Learning, Proceedings of the Fifteenth International Conference (ICML '98)*, pp. 463–471, Morgan Kaufmann, San Francisco, CA, 1998.

[5] W. B. Knox and P. Stone, "Learning non-myopically from human-generated reward," in *Proceedings of the 2013 international conference on Intelligent user interfaces*, pp. 191–202, ACM, 2013.

[6] S. Koenig and Y. Liu, "The interaction of representations and planning objectives for decision-theoretic planning tasks," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 14, no. 4, pp. 303–326, 2002.

[7] S. Mahadevan, "To discount or not to discount in reinforcement learning: A case study comparing r learning and q learning," in *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 164–172, 1994.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. Cambridge Univ Press, 1998.

[9] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, pp. 278–287, 1999.

[10] D. Dolgov and E. Durfee, "Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors," in *In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05*, pp. 1326–1331, 2005.

[11] E. Feinberg and A. Shwartz, "Constrained dynamic programming with two discount factors: applications and an algorithm," *Automatic Control, IEEE Transactions on*, vol. 44, no. 3, pp. 628–631, 1999.

[12] S. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in *ICML*, pp. 656–663, 2003.

[13] S. Singh, R. L. Lewis, and A. G. Barto, "Where do rewards come from," in *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606, 2009.

[14] T. O'Donoghue and M. Rabin, "Doing it now or later," *American Economic Review*, pp. 103–124, 1999.

[15] S. Frederick, G. Loewenstein, and T. O'Donoghue, "Time discounting and time preference: A critical review," *Journal of economic literature*, vol. 40, no. 2, pp. 351–401, 2002.

[16] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Machine Learning*, vol. 84, no. 1-2, pp. 51–80, 2011.

Multi-Objective Markov Decision Processes for Decision Support

Daniel J. Lizotte
Department of Computer Science
Department of Epidemiology & Biostatistics
University of Western Ontario
London, ON N6A 3K7
dlizotte@uwo.ca

Eric B. Laber
Department of Statistics
North Carolina State University
Raliegh, NC 27695
eblaber@ncsu.edu

Abstract

We present a new data analysis framework, Multi-Objective Markov Decision Processes for Decision Support, for developing sequential decision support systems. The framework extends the Multi-Objective Markov Decision Process with the ability to provide support that is tailored to different decision-makers with different preferences about which objectives are most important to them. We present an extension of fitted-Q iteration for multiple objectives that can compute recommended actions in this context; in doing so we identify and address several conceptual and computational challenges. Finally, we demonstrate how our model could be applied to provide decision support for choosing treatments for schizophrenia using data from the Clinical Antipsychotic Trials of Intervention Effectiveness.

Keywords: healthcare, decision support, multi-objective optimization

Acknowledgements

We are deeply indebted to iCORE for their generous support of RLDM2015.

1 Introduction

Markov Decision Processes (MDPs) provide a framework for reasoning about the actions of an autonomous agent in an environment as it strives to achieve long-term success through non-myopic decision making. Operating within this framework, reinforcement learning (RL) methods for finding optimal actions in MDPs hold great promise for using longitudinal data to help humans make better-informed decisions. Indeed, "batch reinforcement learning" methods, e.g. fitted Q-learning [Ernst et al., 2005] are already being used to aid decision-making in diverse areas including medicine, ecology, intelligent tutoring systems, and water reservoir control. Although headway has been made in these application areas, progress is hampered by the fact that many sequential decision *support* problems are not well-modelled by standard Markov Decision Processes. One reason for this is that in most cases, human action selection is driven by multiple competing objectives. In other words, the quality of a policyis not well-captured by a single scalar "reward" or "value." Multi-Objective Markov Decision Processes (MOMDPs) [Roijers et al., 2013] accommodate multiple objectives by allowing vector-valued rewards. "Solving" a MOMDP entails finding a policy or policies that are optimal for a particular *solution concept* which is essentially a partial order on policies. The set of policies that are maximal according to the partial order are considered "optimal" and are indistinguishable according to the chosen solution concept. The widely-used *Pareto optimality* [Ehrgott, 2005] is one example of a solution concept.

We present a new framework, Multi-Objective Markov Decision Processes for Decision Support (MOMDP-DS), for developing decision support systems. Our framework leads us to identify and relax two assumptions inherent in the standard MOMDP solution concepts when used for decision support: i) The assumption that a unique action specified by a policy will be carried out by the decision-maker. ii) The assumption that the preferences of the decision-maker do not change over time. Rather than force the decision maker to select a single policy to follow *a priori*, we prefer to allow her or him to revisit action selection at each decision point in light of new information, both about state and about their own preferences and priorities regarding different outcomes of interest. We can actually accommodate changes in preference over time while still making optimal decisions according to our new solution concepts by introducing the *non-deterministic multi-objective fitted-Q* algorithm. This allows us to perform fitted-Q backups using multiple reward functions. We are particularly motivated by clinical decision-making; therefore we demonstrate the use of our algorithm using data from the Clinical Antipsychotic Trials of Intervention Effectiveness (CATIE).

2 Background

Our new approach extends *Multi-Objective Markov Decision Processes* (MOMDPs) with the goal of providing data-driven decision support. To do so, as an intermediate step we learn a *non-deterministic policy* (NDP) that encodes the set of all non-dominated policies. In this section we review the existing literature on MOMDPs and NDPs.

The most basic definition of a Markov Decision Process is as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $P(s, a, s') = \mathsf{Pr}(s'|s, a)$ gives the probability of a state transition given action and current state, and $R(s, a)$ gives the immediate reward obtained in state s when taking action a. The most common goal of "solving" an MDP is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes $V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^{\,t} R(s_t, a_t)|s_0 = s]$ pointwise for all states. Here, $\mathbb{E}_{\pi}$ indicates that the expectation is taken assuming the state-action trajectories are obtained by following policy π, and $0 < \gamma < 1$.

Like previous work by Lizotte et al. [2012] and by many others, we examine the setting where the typical definition of an MDP is augmented by assuming a D-dimensional reward vector $\mathbf{R}(s_t, a_t)$ is observed at each timestep. We define a *finite-horizon MOMDP* with finite time horizon T as a tuple of state spaces $\mathcal{S}_t$, action spaces $\mathcal{A}_t$, state transition functions $P_t : \mathcal{S}_t \times \mathcal{A}_t \rightarrow \mathbb{P}(\mathcal{S}_{t+1})$ where $\mathbb{P}(\mathcal{S}_{t+1})$ is the space of probability measures on $\mathcal{S}_{t+1}$, and reward functions $\mathbf{R}_t : \mathcal{S}_t \times A_t \rightarrow \mathbb{R}^D$ for $t \in \{1, ..., T\}$. In keeping with the Markov assumption, both $\mathbf{R}_t$ and P_t depend only on the current state and action. In this work we assume finite action sets, but we do *not* assume that state spaces are finite. We define a policy π as a sequence of functions π_t for $t \in \{1, ..., T\}$, where $\pi_t : \mathcal{S}_t \rightarrow \mathcal{A}_t$. The *value* of a policy π is given by $\mathbf{V}^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=1}^{T} \mathbf{R}^t(s_t, a_t)|s_1 = s]$ which is the expected sum of (vector-valued) rewards we achieve by following policy π. We base our method on assessing actions using a *partial order* on their vector of Q-values. Perhaps the most common partial order on vectors comes from the notion of *Pareto-optimality* [Ehrgott, 2005]. For example, an action a is *Pareto-optimal* at a state s_T if for all reward dimensions $d = 1, \ldots, D$ and for all actions a' we have $Q_{T[d]}(s_T, a) \geq Q_{T[d]}(s_T, a')$. We will show that for $t < T$, the problem of deciding which actions are optimal is more complex, but we will still leverage the idea of a partial order.

Milani Fard and Pineau [2011] describe non-deterministic policies for MDPs with a finite state space and a single reward function. Given an MDP with *state space* $\mathcal{S}$ and an *action set* $\mathcal{A}$, an NDP, Π, is a map from the state space to the set $2^{\mathcal{A}} \setminus \varnothing$. One can view the NDP as a compact way of expressing a set of policies that might be executed. Suppose that $\#A = |\Pi(s)|$ is the number of actions provided by the NDP Π at all states. Then the number of policies that are *consistent* with Π, that is, the policies for which $\pi(s) \in \Pi(s)$, is $\#A^{|S|}$. So the NDP Π is a compact encoding of an exponential number of policies. We will make use of this property to encode Pareto-optimal policies.

3 Non-deterministic Fitted-Q for Multiple Objectives (NDFQMO)

Our non-deterministic fitted-Q algorithm for multiple objectives uses finite-horizon, batch data. We present a version that uses linear value function approximation because this model is commonly used by statisticians working in clinical decision support [Shortreed et al., 2011] and because available data often contain continuous-valued patient features (e.g., symptom and side-effect levels, laboratory values, etc.) and outcomes (e.g., symptom scores, body mass index). It is a flexible model because we will not restrict the state features one might use. For learning, we assume a batch of n data trajectories of the form $s_1^i, a_1^i, \mathbf{r}_1^i, s_2^i, a_2^i, \mathbf{r}_2^i, ..., s_T^i, a_T^i, \mathbf{r}_T^i,$ for $i = 1, ..., n$ and $\mathbf{r}_t^i = (r_{t[1]}^i, r_{t[2]}^i, ..., r_{t[D]}^i)^\intercal$ for $t = 1, ..., T$.

At time T, we define the approximate Q-function as the linear least squares fit

$$\text{for } d = 1, \ldots, D \;\; \hat{Q}_{T[d]}(s_T, a_T) = \phi_T(s_T, a_T)^\intercal \hat{\mathbf{w}}_{T[d]}, \;\; \hat{\mathbf{w}}_{T[d]} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(\phi_T(s_T^i, a_T^i)^\intercal \mathbf{w} - r_{T[d]}^i \right)^2 \tag{1}$$

giving the estimated vector-valued expected reward function $\hat{\mathbf{Q}}_T(s_T, a_T) = (\hat{Q}_{T[1]}(s_T, a_T), ..., \hat{Q}_{T[D]}(s_T, a_T))^\intercal$. Here, $\phi_T(s_T, a_T)$ is a feature vector of state and action. Having obtained the $\hat{\mathbf{Q}}_T$ from (1), we construct an NDP Π_T that will give, for each state, the actions one might take at the last time point. For each state s_T at the last time point, each action a_T is associated with a *unique* vector-valued estimated expected reward given by $\hat{\mathbf{Q}}_T(s_T, a_T)$. Thus, we decide which among these vectors is a desirable outcome, and place their corresponding actions into $\Pi_T(s)$. For now we let $\Pi_T(s_T)$ be the set of Pareto-optimal (i.e., non-dominated) actions for state s_T; however, we could use other definitions for $\Pi_T(s_T)$.

For $t < T$, it is only possible to define the expected return of taking an action in a given state if we first decide *which policy will be used to choose future actions.* In standard fitted-Q, for example, one assumes that the future policy is given by $\pi(s) = \arg\max_a \hat{Q}(s, a)$. In the non-deterministic setting, we may know that the future policy belongs to some set, but we do not know *which* policy in the set will be chosen; therefore, we explicitly include the dependence of $\hat{Q}_{t[d]}$ on the choice of future policy:

$$\hat{\mathbf{Q}}_t(s_t, a_t; \pi_{t+1}, ..., \pi_T) = (\hat{Q}_{t[1]}(s_t, a_t; \pi_{t+1}, ..., \pi_T), ..., \hat{Q}_{t[D]}(s_t, a_t; \pi_{t+1}, ..., \pi_T))^\intercal$$

where

$$\text{for } d = 1, \ldots, D \;\; \hat{Q}_{t[d]}(s_t, a_t; \pi_{t+1}, ..., \pi_T) = \phi_t(s_t, a_t)^\intercal \hat{\mathbf{w}}_{t[d]\pi_{t+1}, ..., \pi_T},$$

and

$$\hat{\mathbf{w}}_{t[d]\pi_{t+1}, ..., \pi_T} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(\phi_t(s_t^i, a_t^i)^\intercal \mathbf{w} - (r_t^i + \hat{Q}_{t+1[d]}(s_{t+1}^i, \pi_{t+1}(s_{t+1}^i); \pi_{t+2}, ..., \pi_T)) \right)^2.$$

Figure 1 illustrates what a $\hat{\mathbf{Q}}_{T-1}$ function might look like for a fixed state s_{T-1}, $D = 2$ basis rewards, and twenty different potential future policies π_T. For a fixed value of s_{T-1}, each choice of a_{T-1} and π_T generates a vector-valued estimated Q, $\hat{\mathbf{Q}}_{T-1}(s_{T-1}, a_{T-1}) = (\hat{Q}_{T-1[1]}(s_{T-1}, a_{T-1}; \pi_T), \hat{Q}_{T-1[2]}(s_{T-1}, a_{T-1}; \pi_T))^\intercal$, which we can plot as a point in the plane. We say that such a point represents an expected return that is *achievable* by taking some action in the current state and following it with a fixed sequence of policies until we reach the last time point.

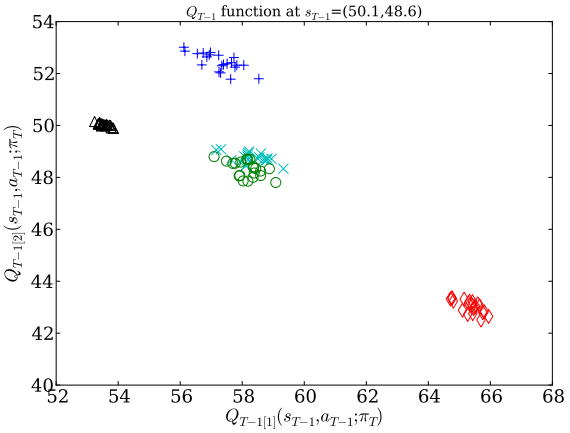


Figure 1: A vector-valued Q-function for a fixed state s_{T-1}. Here, $a_{T-1} \in \mathcal{A} = \{+, \triangle, \bigcirc, \times, \Diamond\}$, and $\pi_T \in \mathcal{P}$, a collection of 20 possible future policies.

Let $\mathcal{Q}_t(a_t, s_t)$ be the set of all expected returns achievable from s_t by taking action a_t and then following some future policy. Given the $\mathcal{Q}_t(a_t, s_t)$ for each a_t, our next task is to use them to define $\Pi_t(s_t)$ for all s_t. Although $\mathcal{Q}_T(s_T, a_T)$ are singletons, for $t < T$ this is not the case, and we must take this into account when defining $\Pi_t(s_t)$. We present two definitions for $\Pi_t(s_t)$ based on a strict partial order $\prec$. (For example $\prec$ may be the Pareto partial order.)

$$\Pi^\forall(s_t) = \{a : \forall \hat{\mathbf{Q}} \in \mathcal{Q}_t(s_t, a) \;\; \nexists a' \neq a, \hat{\mathbf{Q}}' \in \mathcal{Q}_t(s_t, a') \text{ s.t. } \hat{\mathbf{Q}} \prec \hat{\mathbf{Q}}'\}$$
$$\Pi^\exists(s_t) = \{a : \exists \hat{\mathbf{Q}} \in \mathcal{Q}_t(s_t, a) \;\; \nexists a' \neq a, \hat{\mathbf{Q}}' \in \mathcal{Q}_t(s_t, a') \text{ s.t. } \hat{\mathbf{Q}} \prec \hat{\mathbf{Q}}'\}.$$

Under $\Pi^\forall$, action a is included if ***for all*** policies we might follow after choosing a, no other choice of current action and future policy is preferable. $\Pi^\forall$ is appealing in cases where we wish to guard against a naïve decision maker choosing poor sequences of future actions. For the $\mathcal{Q}_{T-1}$ shown in Figure 1, we would have $\Pi^\forall(s_{T-1}) = \{+, \Diamond\}$. The $\triangle$ action is obviously eliminated because any $+$ point dominates every single $\triangle$ point. The $\bigcirc$ and $\times$ actions eliminate each other: There are $\bigcirc$ points that are dominated by $\times$ points, and $\times$ points that are dominated by $\bigcirc$ points. Note that therefore, $\Pi^\forall(s_t)$ may be empty: if our example only contained the $\times$ and $\bigcirc$ actions, we would have $\Pi^\forall(s_{T-1}) = \varnothing$. In

practice we find that $\Pi^{\forall}$ can be very restrictive; we therefore present $\Pi^{\exists}$ as an alternative. Under $\Pi^{\exists}$, action a is included if there is ***at least one*** fixed future policy for which a is not dominated by a value achievable by another $(a', \hat{\mathbf{Q}}')$ pair. Note that $\Pi^{\exists} \supseteq \Pi^{\forall}$, and that because the relation $\hat{\mathbf{Q}} \prec \hat{\mathbf{Q}}'$ is a partial order on a finite set, there must exist at least one maximal element; therefore $\Pi^{\exists}(s_t) \neq \varnothing$. In the Figure 1 example, we have $\Pi^{\exists}(s_{T-1}) = \{+, \diamondsuit, \times\}$; note that $\bigcirc$ is not included because there is always another action that can dominate it if we choose an appropriate future policy. In order to provide increased choice and to ensure we do not generate NDPs with empty action sets, we will use $\Pi^{\exists}$ in our examples. However, because our goal is decision *support*, depending on the application area we may want to present both $\Pi^{\exists}$ and $\Pi^{\forall}$ to a decision maker, or perhaps their set-difference, to give a more complete picture of action consequences.

The next step is to form $\mathcal{Q}_t$ (our set of candidate policies for the current time point) from Π_t (our NDP.) We only consider policies that i) are *consistent* with the learned NDP, and ii) are *representable* using the approximation space chosen for $\hat{Q}$. We define these notions of consistency and representability and show how the set of consistent and representable policies can be efficiently enumerated using mixed integer linear programming. A policy π_t is *consistent* with an NDP Π_t and we write $\pi_t \sqsubset \Pi_t$ if and only if $\pi_t(s_t) \in \Pi_t(s_t)\ \forall s_t \in \mathcal{S}$. We denote the set of all policies consistent with Π_t by $\mathcal{C}(\Pi_t)$. When constructing $\mathcal{Q}_t$, we will only consider future policies that are consistent with the NDPs we have already learned for later time points. This is analogous to fitted-Q in the scalar reward setting, where we estimate the current Q function assuming we will follow the greedy policy of the optimal Q function at later time points. In our setting, there are likely to be multiple policies whose values are considered "optimal." The restriction to consistent policies is not an approximation in the sense that we are only eliminating policies that we assume would never be executed—this is analogous to scalar fitted-Q learning where the only future policy considered is the learned optimal policy. Although we cannot reduce the set of possible future policies to a unique choice as we can for scalar fitted-Q, we can still make significant computational savings. Note that $|\mathcal{C}(\Pi_t)| = \prod_{s_t \in \mathcal{S}^n} |\Pi_t(s_t)|$ where $\mathcal{S}^n$ contains the observed states in our dataset. Because $|\Pi_t(s_t)| \leq |\mathcal{A}|$, we have $|\mathcal{C}(\Pi_t)| \leq |\mathcal{A}|^n$. If Π_t screens out enough actions from enough observed states, restriction to consistent policies can result in a much smaller $\mathcal{Q}_t$. Unfortunately, in the worst case where $\forall s_t\ \Pi_t(s_t) = \mathcal{A}_t$, we have $|\mathcal{C}(\Pi_t)| = |\mathcal{A}|^n$, and if for some fraction η of the n trajectories ($0 < \eta \leq 1$) we have $|\Pi_t(s_t)| \geq 2$, then we have $|\mathcal{C}(\Pi_t)| \in \Omega(2^n)$. Therefore in many interesting cases, including even just the consistent future policies in $\mathcal{Q}_t$ is computationally intractable.

We therefore impose a further restriction on policies in $\mathcal{Q}_t$ that does not eliminate any policies that are optimal according to some scalar reward signal. In scalar fitted-Q, the learned optimal policy is given by $\arg\max_a Q(s,a)$. If the learned Q-functions are linear in some feature space, then the learned optimal policy can be represented by a collection of linear separators that divide feature space into regions where different actions are chosen. This is true for *any* scalar reward signal. Therefore, in the scalar reward case, any future policy that cannot be represented in this way is never considered when computing $\hat{Q}$ for earlier timepoints, no matter what the reward function is. We therefore will "prune away" these consistent but un-representable policies in order to reduce the size of $\mathcal{Q}_t$ by introducing *policy ϕ-consistency*: Given a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^p$, we say a policy π_t is *ϕ-consistent* with a non-deterministic policy Π_t over some dataset with n trajectories, and we write $\pi_t \sqsubset_\phi \Pi_t$, iff $\pi_t(s_t^i) \in \Pi_t(s_t^i)\ \forall i \in 1, ..., n$ and $\exists w \forall s_t^i,\ \pi_t(s_t^i) = \arg\max_a \phi(s_t, a)^\intercal w$. We denote the set of all policies that are ϕ-consistent with Π_t by $\mathcal{C}_\phi(\Pi_t)$. We have two main results about the effect of this pruning, which are given in abbreviated form below without proof. In the following, a *scalarization function* is simply a function that consumes a vector and produces a scalar, and is non-decreasing in every input.

Theorem 1. *Given a linear space of Q-functions based on features ϕ, and scalarization function ρ, there exists a feature-consistent policy π_ϕ for which $\pi_\phi(s) = \arg\max_a \hat{Q}(s,a)$ for all states, where $\hat{Q}$ is learned from rewards given by applying ρ to the $\mathbf{r}_t^i$.*

Theorem 2. *Given a dataset of size n, feature map ϕ, and action set $\mathcal{A}$, there are $O(n^{\dim(\phi)} \cdot |\mathcal{A}|^{2\dim(\phi)})$ feature-consistent policies.*

Theorem 1 implies that for any partial order we care about, for any policy π there is a ϕ-consistent policy that is estimated to be at least as good. Theorem 2 implies that for fixed $|\mathcal{A}|$ and $\dim(\phi)$ there are only polynomially many ϕ-consistent future policies, rather than a potentially exponential number of consistent policies as a function of n. Therefore, by considering only ϕ-consistent future policies, we can ensure that the size of $\mathcal{Q}_{T-1}$ is polynomial in n.

We express the set $\mathcal{C}_\phi(\Pi)$ using a Mixed Integer Program (MIP). To formulate the constraints describing $\mathcal{C}_\phi(\Pi)$, we use *indicator constraints*. Each indicator constraint is associated with a Boolean variable, and is only enforced when that variable is true. We introduce $n \times |\mathcal{A}|$ indicator variables $\alpha_{i,j}$ that indicate whether $\pi(s^i) = j$ or not, adding constraints to ensure that, for each example in our dataset, exactly one action indicator variable is on. We then add indicator constraints to enforce linear separability of the solutions with a margin condition to avoid degeneracy. The resulting policies are recovered from the $\alpha_{i,j}$. Algorithm 1 gives an overview of our non-deterministic fitted-Q algorithm for multiple objectives.

Algorithm 1 NDFQMO

```
Learn Q̂_T = (Q̂_{T[1]}, ..., Q̂_{T[D]}), place in 𝒬_T
for t = T − 1, T − 2, ..., 1 do
  for all s_t^i in the data do
    Generate Π_t^∃(s_t^i) using 𝒬_{t+1}
  for all π_t ∈ 𝒞_φ(Π_t^∃) do
    for all Q̂_{t+1} ∈ 𝒬_{t+1} do
      Learn (Q̂_{[1]t}(·,·,π_t,...), ..., Q̂_{[D]t}(·,·,π_t,...)) using Q̂_{t+1}
      Place (Q̂_{[1]t}(·,·,π_t,...), ..., Q̂_{[D]t}(·,·,π_t,...)) in 𝒬_t
```

4 Empirical Example: CATIE

We illustrate our approach using data from the Clinical Antipsychotic Trials of Intervention Effectiveness (CATIE) study [Stroup, T. S. et al, 2003]. CATIE was an 18-month study of $n = 1460$ patients with two main phases of treatment. Most patients began in "Phase 1," and were randomized to one of five treatmentswith equal probability: olanzapine◊, risperidone×, quetiapine○, ziprasidone+, or perphenazine△. Over time, patients could discontinue their Phase 1 treatment and begin "Phase 2" on a new treatment if needed. Batch RL has been used to analyze this study using a single basis reward [Shortreed et al., 2011] and convex combinations of basis rewards [Lizotte et al., 2012]. We present the treatment recommendations of a non-deterministic fitted-Q analysis using two basis rewards that measure symptom relief and side-effects, the the Positive and Negative Syndrome Scale (PANSS) which measures symptoms, and Body Mass Index (BMI), a measure of obesity. PANSS and BMI are transformed so that higher is better. Our features are the patient's most recent PANSS and most recent BMI, and baseline characteristics.

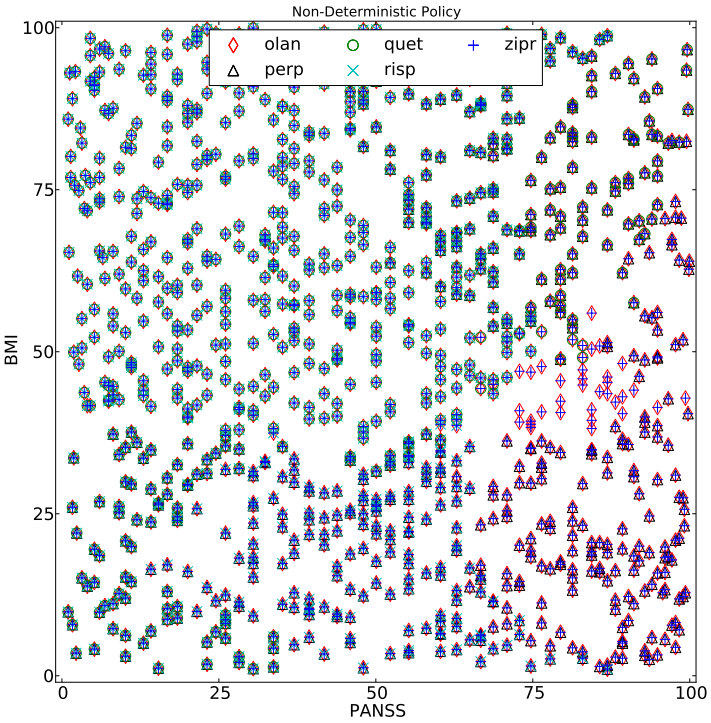


Figure 2: NDP produced by NDFQMO

Figure 2 shows the NDP learned for Phase 1. Each point corresponds to one value of s_1 in our dataset, and at each point is placed a marker for each action recommended by the learned NDP. (Figure 1 is in fact a plot of the Q-function for Phase 1 where $(\text{PANSS}, \text{BMI}) = (50.1, 48.6)$, limited to a $\mathcal{Q}_1$ of size 20 for clarity.) In this NDP, the mean number of choices per state is 3.94, and 82% of states have had one or more actions eliminated while still giving recommendations that allow a decision-maker to produce a Pareto-optimal expected outcome. In comparison, if we eliminate actions using the approach by Lizotte et al. [2012] based on convex reward combinations (not shown) the mean number of choices per state is 2.23, and 100% of states had one or more actions eliminated; these also lead to Pareto-optimal expected outcomes but at the cost of eliminating much more choice. Using ϕ-consistency to reduce computation was critical; in the Phase 2 NDP there are over 10^{124} consistent policies but only 1311 ϕ-consistent policies. Finding these took less than one minute on an Intel Core i7 at 3.4 GHz using CPLEX.

Incorporating uncertainty will be important moving forward. For example, if joint prediction intervals for the returns of an action include a Pareto-optimal point, we could include it in Π even if the Q-value for that action is dominated. This definition of $\Pi(s_t)$ could be incorporated into the framework we have described. Our overarching goal is to expand the toolbox of data analysts by developing new methods for producing decision support systems in very challenging settings. To have maximum impact, decision support must take into account sequential aspects of the problem at hand and at the same time acknowledge the fact that different decision makers have different preferences. Working toward this goal, we have presented a novel approach for learning non-deterministic policies for MDPs with multiple objectives.

References

M. Ehrgott. *Multicriteria Optimization*, chapter 3. Springer, second edition, 2005.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

D. J. Lizotte, M. Bowling, and S. A. Murphy. Linear fitted-Q iteration with multiple reward functions. *Journal of Machine Learning Research*, 13:3253–3295, Nov 2012.

M. Milani Fard and J. Pineau. Non-deterministic policies in markovian decision processes. *Journal of Artificial Intelligence Research*, 40:1–24, 2011.

D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *JAIR*, 48:67–113, 2013.

S. Shortreed, E. B. Laber, D. J. Lizotte, T. S. Stroup, J. Pineau, and S. A. Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine Learning*, 84(1–2):109–136, 2011.

Stroup, T. S. et al. The national institute of mental health clinical antipsychotic trials of intervention effectiveness (CATIE) project: Schizophrenia trial design and protocol development. *Schizophrenia Bulletin*, 29(1), 2003.

Utility-weighted sampling in decisions from experience

Falk Lieder
Helen Wills Neuroscience Institute
University of California at Berkeley, CA
falk.lieder@berkeley.edu

Thomas L. Griffiths
Department of Psychology
University of California at Berkeley, CA
tom_griffiths@berkeley.edu

Ming Hsu
Haas School of Business
University of California at Berkeley, CA
mhsu@haas.berkeley.edu

Abstract

People overweight extreme events in decision-making and overestimate their frequency. Previous theoretical work has shown that this apparently irrational bias could result from utility-weighted sampling–a decision mechanism that makes rational use of limited computational resources (Lieder, Hsu, & Griffiths, 2014). Here, we show that utility-weighted sampling can emerge from a neurally plausible associative learning mechanism. Our model explains the over-weighting of extreme outcomes in repeated decisions from experience (Ludvig, Madan, & Spetch, 2014), as well as the overestimation of their frequency and the underlying memory biases (Madan, Ludvig, & Spetch, 2014). Our results support the conclusion that utility drives probability-weighting by biasing the neural simulation of potential consequences towards extreme values.

Keywords: Decisions from experience; Reinforcement Learning; Bounded Rationality; Bayesian Models; Rational Process Models

Acknowledgements

This work was supported by grant number N00014-13-1-0341 from the Office of Naval Research and grant number RO1 MH098023 from the National Institutes of Health.

1 Introduction

People overweight extreme events in decision-making and overestimate their frequency (Lichtenstein, Slovic, Fischhoff, Layman, & Combs, 1978; Ludvig et al., 2014; Madan et al., 2014). The cognitive bias to overweight extreme events appears irrational, because it violates expected utility theory. Yet, expected utility theory may not be the right metric of rationality for agents with finite computational resources, because it is computationally intractable. Recent behavioral and neural evidence suggests that the brain approximates the solutions to intractable inference and decision problems by Monte-Carlo sampling (Fiser, Berkes, Orbán, & Lengyel, 2010; Vul, Goodman, Griffiths, & Tenenbaum, 2014). This suggests that the brain might approximate expected utilities by an efficient sampling algorithm such as importance sampling (Hammersley & Handscomb, 1964; Geweke, 1989). Every approximation has to trade-off two sources of error: bias and variance. The optimal bias-variance tradeoff of importance sampling overweights extreme events in a manner that resembles people's biases in frequency estimation and risky choice (Lieder et al., 2014). Utility-weighted sampling is a tractable and psychologically plausible approximation to optimal importance sampling that simulates s potential outcomes $o_1^{(a)}, \cdots, o_s^{(a)}$ of an action a with a relative frequency q proportional to the product of their probability $p(o|a)$ and the absolute value of their utility $|u(o)|$:

$$q(o|a) \propto p(o|a) \cdot |u(o)|. \tag{1}$$

The utilities of the simulated outcomes are then combined into an estimate of the expected utility

$$\hat{U}^{(a)} = \frac{1}{\sum_{j=1}^{s} 1/|u(\hat{o}_j^{(a)})|} \sum_{j=1}^{s} \text{sgn}\left(u(\hat{o}_j^{(a)})\right), \tag{2}$$

where sgn is $+1$ for positive and -1 for negative arguments.

We have previously shown that utility-weighted sampling explains the overweighting of extreme events in decisions from description (Lieder et al., 2014). Ludvig et al. (2014) and Madan et al. (2014) demonstrated that in decisions from experience people overweight the more extreme of two outcomes even when both occur with a relative frequency of 50%. These findings speak to utility-weighted sampling and rule out the alternative interpretation that extreme events are over-weighted only because they are rare (Zhang & Maloney, 2012). Here we show that utility-weighted sampling can arise from reward-modulated associative learning during repeated decisions from experience and demonstrate that it is sufficient to explain the findings of Ludvig et al. (2014) and Madan et al. (2014).

2 Reward-modulated associative learning leads to utility-weighted sampling

Utility-weighted sampling can be implemented using a stochastic winner-take-all network (c.f. Nessler, Pfeiffer, Buesing, & Maass, 2013) whose units represent potential outcomes o. The network's inputs represent the alternatives a of the choice and their weights $w_{a,o}$ encode the strength of the associations between the alternatives and the outcomes. These stimulus-reward associations therefore determine the relative frequency with which the network simulates each outcome for each alternative. In this section we propose a learning rule for the weights $w_{a,o}$ that tunes the network to simulate outcomes according to utility-weighted sampling (Equation 1).

Choosing an alternative a and receiving a rewarding outcome o reinforces their association $w_{a,o}$. The association strengthens more the more surprising the outcome is (Courville, Daw, & Touretzky, 2006). Our model captures this effect by updates that are proportional to the absolute value of the reward prediction error $\text{PE}(o)$ as in the successful Pearce-Hall model of classical conditioning (Pearce & Hall, 1980):

$$w_{t+1}(a,o) = \begin{cases} (1-\gamma) \cdot (w_t(a,o) + \alpha \cdot |\text{PE}(o)|) & \text{if } A(t) = a \text{ and } O(t) = o \\ (1-\gamma) \cdot w_t(a,o) & \text{if } A(t) = a \text{ and } O(t) \neq o \end{cases}, \tag{3}$$

where $A(t)$ and $O(t)$ are the chosen alternative and the outcome in trial t, α is the learning rate, and γ is the forgetting rate. The reward prediction error is the difference between the experienced reward $r(o)$ and reward expectancy $\bar{r}$:

$$\text{PE}(o) = r(o_t) - \bar{r}_t. \tag{4}$$

We model the brain's representation of rewards $r(o)$ according to efficient coding Summerfield and Tsetsos (2015):

$$r(o) = \frac{o}{o_t^{\max} - o_t^{\min}} + \varepsilon, \;\; \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2) \tag{5}$$

where ε represents neural noise, and $o_t^{\max}$ and $o_t^{\min}$ track the highest and the lowest received reward and are initialized by 0 and 1 respectively. We assume that the reward expectancy is formed by temporal difference learning:

$$\bar{r}_t = \bar{r}_{t-1} + \eta \cdot (r_t - \bar{r}_{t-1}), \tag{6}$$

where η is the temporal-difference learning rate and the initial reward expectancy $\bar{r}_0$ is a free parameter. We assume that the initial association strengths are zero. This concludes the learning part of our model. To model decision-making we assume that the rate at which units representing alternative a activate units representing outcome o is proportional to their connection strength:

$$P(\hat{O}^{(a)} = o) \propto w_{a,o}. \tag{7}$$

Since the learning rule (Equation 3) increases the weight $w_{a,o}$ with probability $p(o|a)$ by an increment proportional to $|\text{PE}(o)|$ the normalized weight $w_{a,o}/\sum_{o=1}^{n} w_{a,o}$ which determines the probability that outcome o will be simulated (Equation 7) converges to $p(o|a) \cdot |\text{PE(o)}|$. Furthermore, the reward prediction error PE(o) can be interpreted as a utility function for outcomes whose value is relative:

$$u(o) = \text{PE}(o) = \frac{o - \bar{o}}{o^{\max} - o^{\min}} + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2). \tag{8}$$

Therefore the network gradually learns to perform utility-weighted sampling (Equation 1). The simulated outcomes are then read out by a decision network that chooses the alternative with the highest value of the utility estimate defined in Equation 2. Thus, after sufficient learning the simulation network and the decision network jointly perform utility-weighted sampling. The above equations are meant as an abstract specification of network properties rather than the definition of a concrete neural network, but they suggest that the brain can learn to perform utility-weighted sampling. In the following we will refer to this model as utility-weighted learning (UWL).

3 Utility-weighted sampling predicts overweighting of extreme events in decisions from experience

To test whether our model can explain people's risk preferences in decisions from experience and how they change with learning we fitted our model to the block-by-block choice probabilities in the four experiments by Ludvig et al. (2014). In each experiment people make a series of binary choices. For instance, Experiment 1 comprised 5 blocks with 48 choices each. There were a total of four options: a sure gain of $+20$ points, a sure loss of -20 points, a risky gain offering a 50/50 chance of $+40$ or 0, and a risky loss offering a 50/50 chance of 0 or -40 points. In most trials participants either chose between the risky and the sure gain (gain trials) or between the risky and the sure loss (loss trials). After each choice subjects were shown the number of points earned, and they received no additional information about the options. Experiments 2-4 used different outcomes but were otherwise similar.

To account for people's unsystematic errors due to inattention to the task, we extended the model by assuming that people choose randomly with probability p_{random} and use utility-weighted sampling otherwise. The parameter estimates were $s = 1$ samples, learning rate $\alpha = 1$, forgetting rate $\gamma = 0.375$, noise standard deviation $\sigma_\varepsilon = 0.1$, initial reward expectancy $\bar{r}_0 = 3$, TD learning rate $\eta = 0.05$, probability of random choice $p_{\text{random}} = 0.64$. We found that utility-weighted learning captures several qualitative properties of how people's risk preferences changes with experience: Our simulations of Experiments 1-2 captured that people gradually become more risk-averse on loss trials but more risk-seeking on gain trials (Figure 1A). Our simulations of Experiment 3 captured that this effect is reduced when gains and losses are nonextreme in the context in which they occur (Figure 1B), and the simulation of Experiment 4 captured that more experience makes people more risk-seeking when the high outcome is extreme, but more risk-averse when the low outcome is extreme, even if all outcomes are gains or all outcomes are losses (Figure 1C). According to utility-weighted learning the determinant of risk-seeking is that the high outcome is farther away from the learned reward expectancy than the low outcome. The reward expectancy tracks to average across all recent outcomes. Thus, UWL predicts risk seeking when the high outcome is farther away from the average outcome than the low outcome.

Madan et al. (2014) reproduced Experiment 1 from Ludvig et al. (2014) with added memory tests after the choice trials. In addition, Madan et al. (2014) conducted a second experiment in which all outcomes were shifted by +60 points. Since both experiments added a performance-dependent financial bonus, we fitted Madan et al.'s data separately from those of Ludvig, et al. (2014). We modeled people's frequency estimates according to utility-weighted sampling and their answer to the memory recall question by the outcome that occurred most frequently in the decision maker's mental simulations; if two or more outcomes occurred equally frequently one of them was chosen at random. The maximum likelihood parameter estimates indicated increased accuracy motivation: more simulations ($s = 2$), faster learning ($\alpha = 9$), and slower forgetting ($\gamma = 0$). The estimated standard deviation of the noise was $\sigma_\varepsilon = 0.1$, the estimated initial reward expectancy $\bar{r}(0)$ was 7, the estimated rate at which the reward expectancy is updated was 0.5, and the estimated probability of random choice was 0. With these parameters our model captured people's memory biases (see Figure 2) and their relationship with risk seeking: Even though the risky choice generated the moderate outcome (0 points) and the extreme outcome (± 40 points) equally often, for most people the extreme outcome came to mind first (Figure 2B), and their frequency estimates were significantly higher for the extreme loss than for the moderate outcome (Figure 2A). This was not the case for the high gain (+40), because according to the parameter estimates participants entered the experiment with the expectation that outcomes would average 560 points.

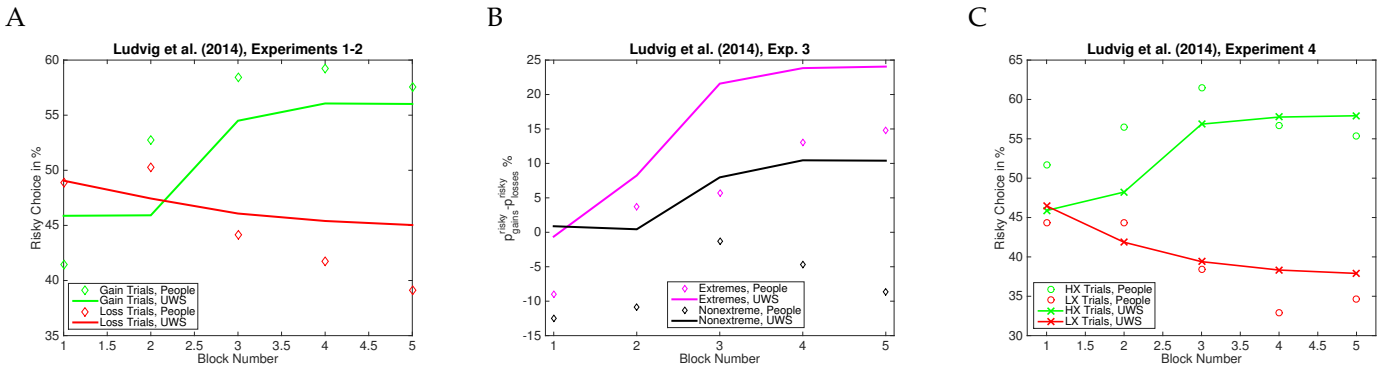


Figure 1: UWL fit to block-by-block risky choice frequency in Experiments 1-4 by Ludvig et al. (2014).

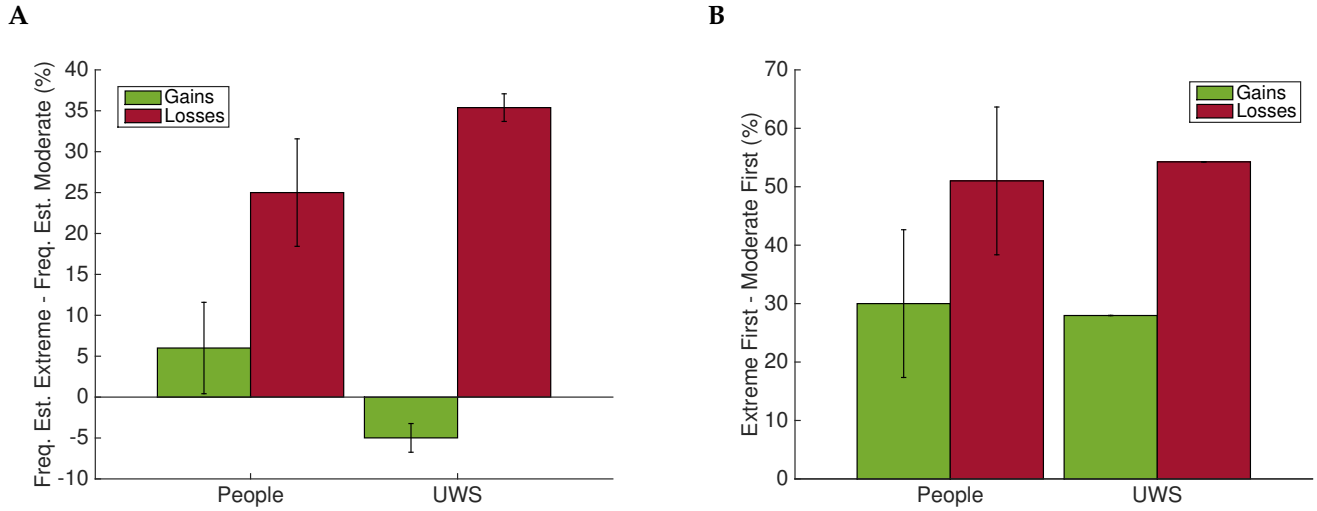


Figure 2: UWL predicts the biased memory recall and frequency estimates observed by Madan et al. (2014). Error bars denote 95% confidence intervals. A: Difference between estimated frequencies of extreme versus moderate outcomes. B: Proportion of people who recalled the extreme outcome first minus proportion who recalled the moderate one first.

In addition, our model correctly predicted that people who recalled the extreme gain first were more risk seeking on gain trials than people who remembered the moderate outcome first ($56.32 \pm 0.24\%$ vs. $50.83 \pm 0.26\%$ risky choices) whereas people who remembered the extreme loss first were less risk seeking on loss trials than people who remembered the moderate outcome first ($31.83\% \pm 0.34\%$ vs. $33.67 \pm 0.34\%$ risky choices). The simulated frequency estimates were significantly correlated with risk seeking: The higher the estimated frequency of the extreme loss the less risk seeking on loss trials ($r = -0.4419, p < 10^{-15}$). The higher the estimated frequency of the extreme gain the more risk seeking on gain trials ($r = 0.23$,$p < 10^{-15}$). Utility-weighted learning also captured that people were more risk seeking when the most recent risky choice in the same context yielded the good outcome than when it yielded the bad outcome: For gain trials UWL predicted 8.6% higher risk seeking after receiving the high gain (+40) than after winning nothing on the previous risky gain trial. Conversely, UWL predicted 6.0% less risk seeking following the large loss (-40) compared to no loss on the previous risky loss trial. Finally, our model captured that all qualitative effects were invariant to adding 60 points to all outcomes so that the same qualitative effects occurred in Experiment 2.

4 Discussion

We have shown that utility-weighted sampling can emerge from reward-modulated associative learning during repeated decisions from experience. Our learning rule assumes that synaptic plasticity is modulated by the absolute value of the reward prediction error (Equation 3). Recent work has discovered a neural correlate of the absolute reward prediction error in the basolateral amygdala (Roesch, Esber, Li, Daw, & Schoenbaum, 2012). This suggests that the proposed learning mechanism could be implemented by neuromodulation of synaptic plasticity in the basolateral amygdala or downstream regions (Dayan, Kakade, & Montague, 2000; McGaugh, McIntyre, & Power, 2002). Our simulation results demonstrate

that utility-weighted sampling is a viable model of decisions from experience. Contrary to the predictions of prospect theory (Kahneman & Tversky, 1979) people over-weight extreme events regardless of their probability (Ludvig et al., 2014). Our model explains this anomaly and provides a quantitative, mechanistic foundation for the *extreme-outcome rule* (Madan et al., 2014) according to which extreme events come to mind first and consequently bias our decisions. Future work will test our model against alternative models of repeated decisions from experience.

The sampling mechanism we are proposing for decisions from experience is slightly different from the one we proposed for decisions from description (Lieder et al., 2014), but this merely reflects the difference in how information is presented in these two paradigms. Future work will evaluate utility-weighted sampling against alternative models on the Technion choice prediction tournament (Erev et al., 2010). Its mechanistic nature allows our model make the following predictions:

1. The rate of stimulus-reward learning is proportional to the extremity of the reward.
2. Incentivising people to consider many possible outcomes should reduce the effect of extreme events, whereas time pressure and cognitive load enhance it.
3. The *probability-weighting function* (Tversky & Kahneman, 1992) depends on the ratio of the outcomes' utilities.
4. Individuals with high reward and loss sensitivity are more susceptible to overestimate extreme events.

Taken together our previous and present work on utility-weighted sampling illustrate how resource-rational analysis (Griffiths, Lieder, & Goodman, 2015) can be used to connect the computational level of analysis to the algorithmic and the implementation level (Marr, 1982).

References

Courville, A., Daw, N., & Touretzky, D. (2006). Bayesian theories of conditioning in a changing world. *Trends Cogn. Sci.*, *10*(7), 294–300.

Dayan, P., Kakade, S., & Montague, P. R. (2000). Learning and selective attention. *Nat. Neurosci.*, *3 Suppl*, 1218–1223.

Erev, I., Ert, E., Roth, A. E., Haruvy, E., Herzog, S. M., Hau, R., … Lebiere, C. (2010). A choice prediction competition: Choices from experience and from description. *J. Behav. Decis. Making*, *23*(1), 15–47.

Fiser, J., Berkes, P., Orbán, G., & Lengyel, M. (2010). Statistically optimal perception and learning: from behavior to neural representations. *Trends Cogn. Sci.*, *14*(3), 119–130.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, *57*(6), 1317–1339.

Griffiths, T. L., Lieder, F., & Goodman, N. D. (2015). Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, *7*(2), 217-229.

Hammersley, D. C., & Handscomb, J. M. (1964). *Monte Carlo methods.* London: Methuen & Co Ltd.

Kahneman, D., & Tversky, A. (1979). Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, *47*(2), 263–291.

Lichtenstein, S., Slovic, P., Fischhoff, B., Layman, M., & Combs, B. (1978). Judged frequency of lethal events. *J. Exp. Psychol. Hum. Learn. Mem.*, *4*(6), 551.

Lieder, F., Hsu, M., & Griffiths, T. L. (2014). The high availability of extreme events serves resource-rational decision-making. In *Proc. 36th ann. conf. cognitive science society.* Austin, TX: Cognitive Science Society.

Ludvig, E. A., Madan, C. R., & Spetch, M. L. (2014). Extreme outcomes sway risky decisions from experience. *J. Behav. Dec. Making*, *27*(2), 146–156.

Madan, C. R., Ludvig, E. A., & Spetch, M. L. (2014). Remembering the best and worst of times: Memories for extreme outcomes bias risky decisions. *Psychonomic bulletin & review*, *21*(3), 629–636.

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. San Francisco: W. H. Freeman and Company.

McGaugh, J., McIntyre, C. K., & Power, A. E. (2002). Amygdala modulation of memory consolidation: Interaction with other brain systems. *Neurobiology of Learning and Memory*, *78*(3), 539–552.

Nessler, B., Pfeiffer, M., Buesing, L., & Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput Biol*, *9*(4), e1003037.

Pearce, J. M., & Hall, G. (1980). A model for Pavlovian learning: variations in the effectiveness of conditioned but not of unconditioned stimuli. *Psychological review*, *87*(6), 532.

Roesch, M. R., Esber, G. R., Li, J., Daw, N. D., & Schoenbaum, G. (2012). Surprise! Neural correlates of Pearce-Hall and Rescorla-Wagner coexist within the brain. *European Journal of Neuroscience*, *35*(7), 1190–1200.

Summerfield, C., & Tsetsos, K. (2015). Do humans make good decisions? *Trends in cognitive sciences*, *19*(1), 27–34.

Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *J. Risk Uncertain.*, *5*(4), 297–323. doi: doi:10.1007/bf00122574

Vul, E., Goodman, N. D., Griffiths, T. L., & Tenenbaum, J. B. (2014). One and done? Optimal decisions from very few samples. *Cognitive Sci.*, 1–39.

Zhang, H., & Maloney, L. T. (2012). Ubiquitous log odds: A common representation of probability and frequency distortion in perception, action, and cognition. *Frontiers in Neuroscience*, *6*.

Bayesian Learning for Safe High-Speed Navigation in Unknown Environments

Charles Richter, William Vega-Brown, and Nicholas Roy
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{car,wrvb,nickroy}@mit.edu

Abstract

The focus of this work is to develop a planner for high-speed navigation in unknown environments, for instance locating a specified goal in an unknown building in minimum time or flying as fast as possible through an unmapped forest. We model this problem as a POMDP and discuss why it is so difficult even under the assumption of noiseless dynamics and observations. We then describe our method of predicting probabilities of collision as a way to approximate the POMDP solution. We employ a Bayesian non-parametric learning algorithm to predict probabilities of collision associated with different planning scenarios, and select trajectories in a receding-horizon fashion that minimize cost in expectation with respect to those probabilities. We also describe the training procedure for our learning algorithm and draw the similarities between our approach and batch, model-based reinforcement learning. We show two principal results. First, we show that by using a learned model of collision probability, our robot can navigate significantly faster in certain environments than a robot that enforces absolute safety guarantees, provided that it has access to training data from similar environments. Second, leveraging the Bayesian nature of our learning algorithm, we show that in situations where the robot does not have any relevant training data to draw upon, it seamlessly and automatically reverts to a prior estimate of collision probability that keeps it safe.

Keywords: autonomous navigation, unknown environment, Bayesian non-parametric learning, batch model-based reinforcement learning, POMDP

1 Introduction

Mobile robots and autonomous vehicles offer solutions to a wide range of practical problems, including information gathering, industrial fabrication and assembly, home or workplace service, and many more. In each one of these cases, a robot must move about in a space that may be previously unmapped, or a prior map may be out of date due to rearrangement of objects. In either case, the level of prior map knowledge may not be sufficient to plan high-speed collision-free trajectories. Still, we would like robots to move quickly so that they can complete tasks in a timely manner. We would like to close the speed gap with human pilots and fast moving animals that can navigate at high speeds based on their learned or intuitive understanding of the way their actions and perceptual capabilities work in familiar environments.

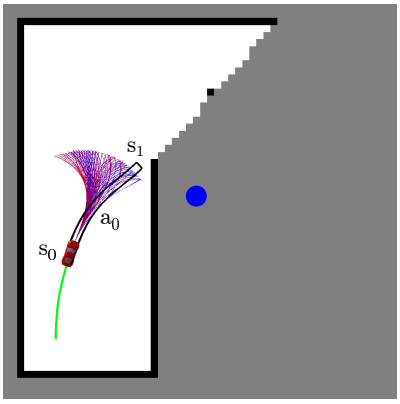


Figure 1: The set of actions a_0 for a car in state s_0 is shown in an example planning scenario. The goal (blue) is located in an unobserved portion of the map (gray).

1.1 POMDP formulation

We wish to control a dynamic vehicle, such as a car, airplane, or quadrotor from a start location through an unknown environment to a specified goal in minimum expected time, where the expectation is taken with respect to an unknown map. While solving this problem exactly as a partially observable Markov decision process (POMDP) is completely intractable, the POMDP formalism is useful for specifying the planning task. The following POMDP tuple, (S, A, T, R, O, Ω), applies to an RC car equipped with a planar LIDAR being used for this research:

States S: $\{Q \times M\}$, where:

- Q is the set of continuous-valued vehicle configurations: $q = [x, y, \psi, k, v] \in \mathbb{R}^5$.
- M is the set of n-cell occupancy maps: $m = [m_0, m_1, \ldots, m_n] \in \{0, 1\}^n$.

For a given problem instance, the true underlying map, m, is fixed for all time, while the configuration, q, changes at each time step as the robot moves. We assume that q is fully observable, while m is partially observable since only a subset of the map cells can be observed by the sensor from a given location.

Actions A: A pre-computed discrete set of trajectories of specified length, which roughly span the vehicle's maneuvering capabilities. Figure 1 shows an example of action set A. All actions are the same length, but differ in their time duration as a function of their speeds.

We define two functions here for convenience. The deterministic collision function $C(s, a) : S \times A \mapsto \{0, 1\}$ indicates whether taking action a from state s would result in a collision. The deterministic state-transition function $F(s, a) : S \times A \mapsto S$ returns the future state reached by taking action a from state s. In this state transition, the map portion of the state remains fixed at its true value. If a collision would occur along trajectory a (i.e., if $C(s, a) = 1$), then $F(s, a)$ clamps the future vehicle configuration to its last feasible position and orientation along trajectory a and sets the velocity to zero.

Observations Ω: The set of LIDAR scans containing l ranges, r_i, to the nearest occupied map cells around the agent, $o = [r_1, r_2, \ldots, r_l] \in \mathbb{R}^l_+$.

Conditional Transition Probabilities T: We assume deterministic vehicle dynamics and a fixed map, $P(s_{t+1}|s_t, a_t) = 1$ for $s_{t+1} = F(s_t, a_t)$ and 0 otherwise.

Conditional Observation Probabilities O: We assume a noiseless sensor observing the environment and vehicle configuration, so $P(o_{t+1}|s_{t+1}, a_t) = 1$ for the LIDAR scan corresponding to the map geometry visible from state s_{t+1}, including a perfect measurement of q_{t+1}, and 0 otherwise.

Cost Function R: We use a minimum-time cost function $R(s, a)$, which returns the time duration of an action $J_a(a)$ if the action succeeds without collision, and adds an additional collision penalty J_c if the action results in a collision. $R(s, a) = J_a(a)$ if $C(s, a) = 0$, and $R(s, a) = J_a(a) + J_c$ if $C(s, a) = 1$.

1.2 Belief space

In a general belief-state MDP, the agent maintains a belief over its state $b(s) = P(q, m)$. We use b_t to denote the belief at time t and m_t to denote the belief over maps at t. Since q is fully observable in our problem, the belief really represents the agent's belief over which of the exponentially many possible maps it is in. Let $\rho(b, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a)$ be a belief-state cost function. The optimal value (or cost-to-go) for a belief can in principle be computed using the Bellman equation:

$$V^*(b_t) = \min_{a_t} \left\{ \rho(b_t, a_t) + \sum_{s_t} b(s_t) \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \sum_{o_{t+1}} P(o_{t+1}|s_{t+1}, a_t) V^*(s_{t+1}) \right\} \quad (1)$$

In the summation over s_t, equation (1) is effectively taking an expectation over maps where the probability applied to each possible map is contained within $b(s_t)$. The summation over o_{t+1} performs a state estimation update to compute the distribution over future states given the current belief b_t and an action a_t. If our prior belief, b_t, assigns uniform probability to all possible maps, then this state estimation update serves to eliminate those maps that are not consistent with the current observation and raise the uniform probability of the remaining possible maps. If our prior belief assigns high probability to realistic maps containing common structures such as hallways, rooms, doors, etc., and low probability to unrealistic maps, then this state estimation update would help us infer useful information about regions of the map that have not been directly observed yet. For instance, it might infer that the straight parallel walls of a hallway are likely to continue to be straight parallel walls out into the unknown regions of the map. All of this prior knowledge about likely environments enters the problem through the agent's initial belief, b_0, which we must somehow provide.

1.3 Missing information: The central challenge of this work

Unfortunately, learning an accurate distribution over the space of all real-world environments the robot may encounter would be extremely difficult. This difficulty results from the high dimensionality ($\approx 10^6$) of building-sized occupancy grid maps, the strong assumption of independence between map cells, and the richness of man-made and natural spaces which resist a more compact parameterization. Without significant modeling effort, or restriction of the problem domain to specific environments, the best prior we can reasonably provide is to say that every map is equally likely. Of course, this prior is completely unhelpful for planning, and prevents the agent from exploiting intuitive knowledge of "typical" environments to navigate faster. Our solution must compensate for this missing knowledge.

Our approach is to learn a specific function that enables the agent to drive at high speed *as if* it had an accurate prior over environments enabling it to make reasonable inferences about the unobserved environment. We approximate the expected future reward that is computed in the right-hand end of equation (1) using a learned function, $f_c(\phi(b_t, a_t))$, that estimates the probability that a collision will occur with some unforeseen obstacle in the future, given some features ϕ of a planning scenario. This learned function eliminates the need for expensive summations over states and observations and an explicit prior over environments. The relevant information that would be contained within a prior over environments is represented instead implicitly through the training data for the learning algorithm.

2 Predicting Future Collisions

In this section, we focus on learning to predict the probability of collision associated with a given planning scenario. In practice, it is common for planners to avoid collisions by rejecting actions that commit the robot to entering unknown space, implying an assumption that driving into unknown regions of the map always carries a high probability of collision. However, our central claim in this line of research is that this assumption may not be correct. In many cases, the agent can indeed plan to drive at high speeds into unknown regions of the map without significant risk if it is equipped with prior experience in similar environments [1]. We summarize this experience as a learned function:

$$f_c(\phi(b_t, a_t)) \approx P(\text{"collision is inevitable if we execute } a_t \text{ from belief } b_t\text{"}) \qquad (2)$$

where $\phi(b_t, a_t)$ is a vector of features summarizing the agent's current belief (particularly the map estimate) and the selected action. Relevant features for this prediction might include vehicle speed along a_t, distance to the map frontier, some measure of clutter, free space ahead, etc.

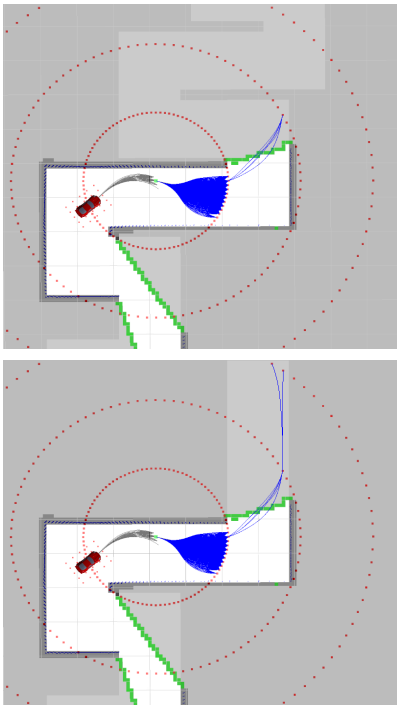


Figure 2: Examples of "collision" and "non-collision" training events.

To predict collision probabilities, we collect a training data set $\mathcal{D} = \{(\phi_1, y_1), \ldots, (\phi_N, y_N)\}$ offline in a manner reminiscent of batch model-based reinforcement learning. Labels y_i are binary indicators ("collision," "non-collision") associated with a planning scenarios described as points ϕ_i in feature space. We generate each data point by randomly sampling a feasible configuration q_t within a training map, and simulating the sensor from q_t to generate a belief b_t. We then randomly select among the reachable configurations, q_{t+1}, and run a (resolution) complete planner to determine whether there exists any feasible trajectory from q_{t+1} that avoids collision with the true underlying map to some horizon (several times longer than the horizon of the initial action a_t). If there exists a feasible trajectory, $y_{\text{new}} = 0$, otherwise $y_{\text{new}} = 1$. Finally, we compute the features $\phi_{\text{new}}(b_t, a_t)$ describing this planning scenario and insert $(\phi_{\text{new}}, y_{\text{new}})$ into $\mathcal{D}$. Figure 2 illustrates an example of "collision" and "non-collision" points being generated.

We use a non-parametric Bayesian inference model developed by Vega-Brown, et al., which generalizes local kernel estimation to the context of Bayesian inference for exponential family distributions [2]. In our case, we consider collision to be a Bernoulli distributed random event with beta-distributed parameter $\theta \sim Beta(\alpha, \beta)$, where α and β are prior

pseudo-counts of collision and non-collision events, respectively. Using the inference model from Vega-Brown, et al., we can efficiently compute the posterior probability of collision given a query point ϕ and the observed training data set $\mathcal{D}$:

$$f_c(\phi) = P(y = \text{“collision”}|\phi, \mathcal{D}) = \frac{\alpha(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i) y_i}{\alpha(\phi) + \beta(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i)}, \tag{3}$$

where $k(\phi, \phi_i)$ is a kernel function reflecting proximity in feature space between our query point ϕ and each other data point in $\mathcal{D}$. We write prior pseudo-counts as functions $\alpha(\phi)$ and $\beta(\phi)$, since they may vary across the feature space. We use the kernel sum $N_{\text{eff.}} = \sum_{i=1}^{N} k(\phi, \phi_i)$, i.e., the effective number of training data points near ϕ, as a measure of data density. The prior also contributes $N_{\text{pr.}}$ pseudo data points to each prediction, where $N_{\text{pr.}} = \alpha(\phi) + \beta(\phi)$. The ratio $N_{\text{eff.}}/(N_{\text{eff.}} + N_{\text{pr.}})$ determines the relative influence of the training data and the prior in each prediction. We use $N_{\text{pr.}} = 10$, and typically when planning in an environment for which we have training data, $N_{\text{eff.}} \approx 500$.

2.1 Planning with learned probabilities of collision

In this section, we describe the specific approximations we make that reduce equation (1) to our proposed objective function. First, we restrict the planner to disregard future environmental measurements since we have no meaningful way yet to predict what the most likely future measurements might be given our uniform prior over maps. The planner's belief of the map thus remains fixed to its current value b_t at planning time, and this eliminates the summation over o_{t+1}. As map knowledge decays with distance ahead of the vehicle, so does the information available for collision prediction. Therefore, rather than attempting to predict collisions deep in the unknown regions of the map, along a long sequence of actions, we simplify the problem by lumping all future collision probability beyond a_t into a single prediction f_c, independent of the future action sequence $(a_{t+1}, a_{t+2}, \dots)$.

We also assume that no collisions occur during the chosen action a_t, since the agent can nearly always perceive obstacles in its immediate vicinity and perform conventional collision checking. This assumption allows us to replace $\rho(b_t, a_t)$ with $J_a(a_t)$. Finally, we assume that the remaining cost-to-go after a_t, given the map knowledge m_t contained in belief b_t, is well approximated by a heuristic function $h(q_{t+1}, m_t)$. This heuristic, coupled with our learned probability function, provides a simple approximation to the expected future reward beyond action a_t that would be impossible to compute without an explicit prior over environments, and intractable even with one. Having made these approximations, equation (1) reduces to our proposed objective function:

$$\text{Expected Cost Objective Function:} \quad V^*(b_t) \approx \min_{a_t} \left\{ J_a(a_t) + f_c(\phi(b_t, a_t)) J_c + h(q_{t+1}, m_t) \right\}. \tag{4}$$

At each time step, in receding horizon fashion, we select the action a_t^* minimizing this objective. We begin to execute a_t^* while re-planning. By re-planning at a high rate, we incorporate new sensor data into the belief as it is obtained.

2.2 Relevance of training data across environment types

Machine learning algorithms are designed to make good predictions for query points near their training data. For some algorithms, predictions that extrapolate beyond the region of training data may be arbitrarily bad. For navigation tasks, we want our planner to recognize when it has left the region of feature space for which it has training data, and automatically revert to safe behaviors. For instance, if the agent moves from a well-known environment type into one it has never seen before, we want it to stay safe. Fortunately, the inference model of equation (3) enables this capability.

The Bayesian nature of our inference model enables us to inject prior knowledge that will keep the planner safe when there is little relevant data. If we query a feature point in a region of high data density ($N_{\text{eff.}} \gg N_{\text{pr.}}$), $f_c(\phi)$ will tend to a local weighted average of neighbors and the prior will have little effect. However, if we query a point far from the training data ($N_{\text{eff.}} \ll N_{\text{pr.}}$), the prior will dominate the prediction. By specifying priors $\alpha(\phi)$ and $\beta(\phi)$ that are functions of our features, we can endow the planner with all of our own (perhaps conservative) domain knowledge about which regions of feature space are safe and which are dangerous. In this work, we have designed our prior functions $\alpha(\phi)$ and $\beta(\phi)$ such that $P(\text{“collision”}) = \alpha(\phi)/(\alpha(\phi) + \beta(\phi)) = 0$ if there exists enough free space for the robot to come safely to a stop from its current velocity, and $P(\text{“collision”}) = \alpha(\phi)/(\alpha(\phi) + \beta(\phi)) = 1$ otherwise. Therefore, as $N_{\text{eff.}}$ drops to zero, this stopping distance safety constraint becomes active.

3 Results

In this section, we present simulation results in two different types of environments, the “Markov Hallway” and the “Poisson Forest”. We use generative distributions for these map types to sample random environments. A Markov chain enables us to sample hallways with a specified width and turn frequency, and a Poisson process enables us to sample 2D forests with an average obstacle rate. Figure 4 shows an environment with both hallway and forest segments.

To measure the performance of our planner and the benefit of our learned model, we compare against a conservative baseline planner that guarantees absolute safety. This baseline planner only considers trajectories that lie within the

known free space and it ensures that it always has room to stop before hitting an obstacle or unknown map cell. This planner is intended to navigate as fast as is possible without the use of domain knowledge or hand-coded behaviors to increase speed, and represents the solution we might select if we did not have a learned model of collision probability.

Figure 3 shows the performance of the planner in 25 randomly sampled hallway environments over a range of J_c values. The blue plots show performance using dense training data collected in a hallway environment drawn from the same distribution, plus prior knowledge. The red plots show performance of the planner making predictions based on the data alone, using no prior. The black plots show the performance with the prior only, and no data. The left-most plot shows that the for low values of J_c all planners crash in every trial, but become safer as J_c is increased. Above $J_c = 0.25$, all planners succeed in reaching the goal without collision 100% of the time. The middle plot shows velocity normalized by the speed of baseline planner. Finally, the right-most plot illustrates average violation of the stopping-distance constraint. For $J_c = 0.25$, the planners with training data violate their stopping-distance constraints by about $5.75m$ on average, indicating that they are indeed taking risks to travel as fast as possible while maintaining an empirical success rate of 100%. In practice, we must pick a value of J_c to determine how aggressive and risky the planner will be. If we pick $J_c = 0.25$, we can expect to succeed in every trial while navigating 1.8 times faster than baseline.

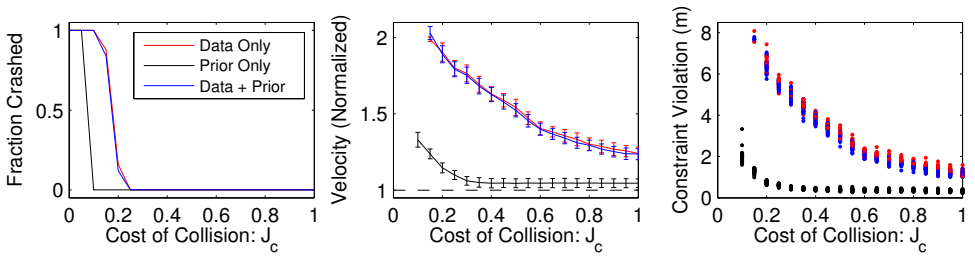


Figure 3: Simulation results using our learned model with dense training data combined with the prior (blue), using the data alone with no prior (red), and using the prior alone without any data (black). The prior alone is enough to keep the robot safe in the absence of data, but having training data improves performance.

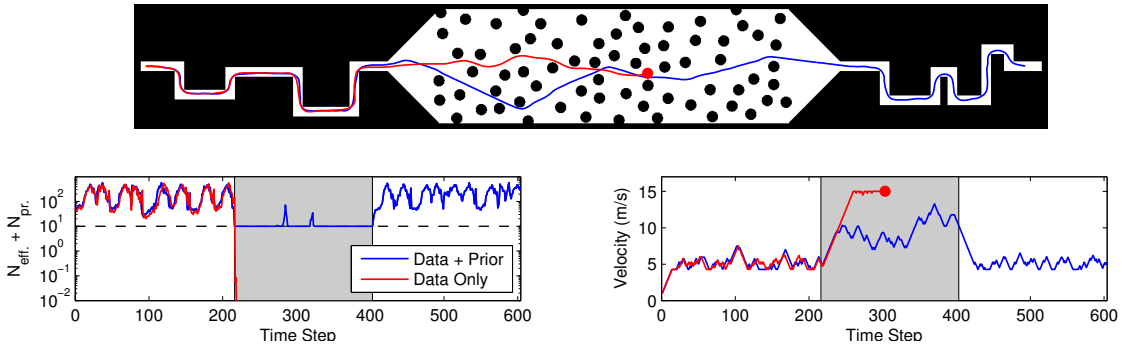


Figure 4: Simulation results from a hybrid hallway-forest map. One planner uses hallway data combined with the prior (blue), and another uses hallway data alone with no prior (red). Without a safe prior, the robot crashes in the forest.

Figure 4 illustrates the use of a safe prior when transitioning from a known environment (hallway) to an unknown one where no data are available (forest). With the prior, the effective number of data points drops to $N_{\text{pr.}} = 10$ when the agent enters the forest (gray region in left and right plots) and the planner is guided by the information in the prior. If the planner has no data and no prior, the effective data density drops to zero and it is unable to distinguish between safe and risky behaviors. Therefore it accelerates to full speed resulting in a crash (red dot). In 20 trials of random hallway-forest maps, 100% succeeded with the prior, and 100% crashed without the prior.

References

[1] Charles Richter, John Ware, and Nicholas Roy. High-speed autonomous navigation of unknown environments using learned probabilities of collision. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[2] William Vega-Brown, Marek Doniec, and Nicholas Roy. Nonparametric bayesian inference on multivariate exponential families. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2014.

Open-Ended Learning of Skills in Robots: Insights from Looking at the Brain

Gianluca Baldassarre, Francesco Mannella, Vieri Giuliano Santucci, Valerio Sperati,
Daniele Caligiore, Emilio Cartoni, Bruno Castro da Silva*, Marco Mirolli
Istituto di Scienze e Tecnologie della Cognizione,
Consiglio Nazionale delle Ricerche,
Roma, Italy
{gianluca.baldassarre, francesco.mannella, vieri.santucci, valerio.sperati,
daniele.caligiore, emilio.cartoni, marco.mirolli}@istc.cnr.it
*Visiting Researcher, bsilva@cs.umass.edu

Abstract

The capacity to learn an increasing number of flexible sensorimotor skills in an open-ended autonomous fashion is one of the most remarkable features of human intelligence. How can we design robot controllers that exhibit the same capability? We start from the idea that to accomplish this objective one needs to design a complex architecture formed by multiple components, rather than single algorithms. The decisions at the high-level are critical for the specification of the single components and their interplay, and hence for the overall success of the system. Here we look at the brain architecture and functioning possibly underlying open-ended development in humans and claim that it suggests two possible principles that can be exploited to build open-ended learning robots: (a) The sensorimotor hierarchy underlying motor behaviour should have three specific levels of organisation rather than a continuously graded granularity; (b) Intrinsic motivations should guide the formation of specific goals, and regulate the skill learning processes pivoting on them, rather than guiding learning processes at fine spatial and temporal scales. Here we draw from the biology to support these principles and present our past and current work implementing and testing their utility for open-ended learning robots.

Keywords: Multiple skill learning, intrinsic motivations,
hierarchical and modular reinforcement learning,
brain macro-architecture, autonomous humanoid learning robots.

Acknowledgements

This research has received funds from the European Commission under the 7th Framework Programme (FP7/2007-2013), ICT Challenge 2 "Cognitive Systems and Robotics", project "IM-CLeVeR - Intrinsically Motivated Cumulative Learning Versatile Robots", grant agreement no. ICT-IP-231722, www.im-clever.eu. We thank Andrew Barto for giving us important suggestions on how to improve some of the ideas presented in the paper and their presentation.

1 Introduction

The capacity of autonomously learning an increasing number of flexible sensorimotor skills in an open-ended fashion is one of the most remarkable features of human intelligence. This capacity relies on key features of the brain's architecture, functioning, and learning, such as a hierarchical structure capable of storing multiple sensorimotor skills at different levels of abstraction, and intrinsic motivation systems driving learning in an autonomous open-ended fashion.

The construction of robots capable of truly open-ended learning would not only be a milestone for artificial intelligence but also a fundamental technological achievement. Open-ended robots could be placed in specific environments and asked to autonomously acquire a rich repertoire of skills without the need for external intervention. The users of the robots could later require such robots to carry out specific tasks relevant to them: the robots could accomplish these tasks with little or no additional learning by relying on the skills previously acquired through autonomous exploration. This would be useful, for example, for service robots having to act in environments posing challenges unforeseen at design time. Open-ended leaning curious robots, generating a continuous flow of novel/surprising behaviours and situations, would also be useful for educational/entertaining applications.

The objective of this abstract is to propose two insights that have been gained by looking at how the human brain *possibly* implements open ended learning. We say 'possibly' since unfortunately we still lack neuroscientific research directly addressing the problem of open-ended learning. These insights relate to two principles that can be followed to design an architecture for open-ended learning robots. We discuss the two principles in the next sections, first mentioning the biology that suggested them, then expanding the implications for the architecture, and finally illustrating some initial implementations of them and their current and future possible extensions.

2 The sensorimotor structure of the architecture should pivot on three levels

Biology. Looking at the primate brain, it appears that the flexibility of their behaviour relies on a three-level organisation. First, information from the musculoskeletal system is processed within the somatosensory cortex to guide movement performance through its closed loops with the primary motor cortex. One important recent insight about the functioning of this system is that these circuits have an inherently dynamic nature and produce sophisticated movement trajectories as opposed to the classic (opposing) views of motor cortex as encoding equilibrium points (desired postures) or low-level dynamic aspects of movements (e.g., muscle forces) [Graziano, 2011]. Second, movement trajectories are modulated by premotor cortex which possibly encodes a repertoire of actions (e.g., 'reach', 'fine grap', 'power grasp', etc.) in terms of 'proximal goals' (i.e., immediate visual and proprioceptive effects) [Gallese and Goldman, 1998] formed on the basis of information on proprioception and object size, position, and orientation. Overall, data on the premotor/motor cortex functioning suggest that the brain learns and plans movement trajectories together with their dynamic implementation (control). Third, the prefrontal cortex forms higher-level goals at multiple levels of abstraction (e.g., 'eating' or 'hitting a target with a stone') [Miller and Cohen, 2001] and through these triggers specific proximal-goal/action sequences. These cortical areas, characterised by associative (Hebbian) learning processes, form segregated loops with basal-ganglia, which learn to select their contents by trial-and-error learning processes, and cerebellum, which learns by supervised learning processes [Middleton and Strick, 2000]. With practice, action and action-sequence performance becomes habitual (i.e., directly triggered by stimuli) and under the control of the first two levels. However, the third level comes into play again when expected action outcomes are not accomplished.

Robots. Within machine learning, the general framework to implement the autonomous learning of hierarchical skills is the reinforcement learning (RL) option framework (OF) [Sutton et al., 1999]. An option is formed by a policy, an initiation set, and a termination function. Critically, the policy of an option can recall primitive actions and also other options. This means that the OF allows the creation of options (skills) with any level of granularity along a continuum.

The OF has been applied with success to grid/discrete setups [Barto and Mahadevan, 2003] but has not taken off in robot control. The introduction of 'dynamic movement primitives' (DMPs) [Ijspeert et al., 2002] has marked an important advance in this field. DMPs are dynamical models that can produce discrete or rhythmic movements. Some parameters of a DMP regulate the trajectory and dynamics of a movement, while metaparameters establish the trajectory initial point, final point ('goal'), and execution speed. RL techniques can search for these parameters, thus optimising the movement trajectory and dynamics [Kober and Peters, 2009]. Notwithstanding the success of these approaches for learning single tasks, few systems can learn to compose multiple movement primitives/DMPs to form more complex behaviours and have various limitations, so this is largely an open problem (e.g., Konidaris et al., 2011, Stulp et al., 2012).

The biological evidence mentioned above suggests organising an open-ended learning architectures as follows. First, the architecture should not be organised at a 'continuous' granularity, as implicitly suggested by the OF, but rather be *organised at three levels*. The lower level should be based on dynamic devices, such as DMPs, acquiring with RL the movement trajectories and dynamics solving tasks (interestingly, alongside control theory, initial inspiration on DMPs is rooted on organisms' central pattern generators, Ijspeert and Kodjabachian, 1999). The intermediate level should

select and trigger different DMPs (e.g., controlling different actuators) and also furnish high-level parameters to them (e.g., initial/final postures depending on target position, movement speed, etc.). The granularity of encoded movements should be at the level of 'grasp', 'reach', etc. Learning could be based on RL and associative/supervised learning. The third level should encode higher-level goals. In contrast to the other two levels, this level should be able to encode goals with different levels of granularity, similar to what is done in the OF (which, however, does not propose specific mechanisms to manage the learning and management of recursive option execution).

Some of our models implement the first level of the architecture and in part its second level. In Castro da Silva et al. [2014] we proposed a humanoid robot controller that learns to map the position of a target in space to suitable parameters of a DMP used to throw a ball to the target. This mapping is based on supervised learning techniques and methods to identify low-dimensional manifolds, within the DMP parameter space, corresponding to policy parameter clusters. In Santucci et al. [2014] we showed that reinforcement learning processes are important for learning to select, through goals, different 'expert controllers' sending their output to different sets of actuators, e.g., to the left or right arm of a robot. In the future, the second level of these systems could be extended by using information on the initial posture of the robot, and on the world state, to set the meta-parameters of the lower-level dynamical skills. This would augment its on-line flexibility. Moreover, the third level of the architecture could be implemented on the basis of RL mechanisms operating on abstract representations of lower-level skills capturing their overall states such as 'not applicable', 'in execution', 'goal accomplished' [Hart and Grupen, 2011, Konidaris et al., 2011].

3 Intrinsic motivations should lead to the self-generation of goals and guide the acquisition of skills directed to them

Biology. The hierarchical brain system described above is regulated by motivational systems playing at least two roles: (a) they motivate the selection of particular actions and goals; (b) they modulate learning, e.g., by triggering the neuromodulator dopamine that regulates trial-and-error learning in basal ganglia. Basic motivations are related to the attainment of resources important for homeostatic regulations of the body (e.g., food, water, warm, physical integrity) [Mirolli et al., 2010]. These are also called 'extrinsic motivations' (EMs) to distinguish them from 'intrinsic motivations' (IMs). IMs, instead, are related to the performance of actions 'for their own sake' [Berlyne, 1966], i.e., without the immediate biological function of the attainment of external material resources relevant for biological fitness. IMs, related to things such as curiosity and exploration, have been proposed to have the biological function of driving the learning of knowledge and skills that are used only later to obtain useful resources, resources that are possibly not present at the moment of the learning [Singh et al., 2010, Baldassarre, 2011]. In neuroscience, issues relevant for IMs are studied with different research objectives: we consider two of them which are most relevant for open-ended learning. First, research on memory and the hippocampus has shown how this structure strongly responds to novel items (e.g., objects) and novel associations between familiar items [Kumaran and Maguire, 2007]. On this basis, the hippocampus regulates motivation and learning in various parts of brain through dopamine [Lisman and Grace, 2005], in particular at the level of the basal ganglia/prefrontal cortex system dealing with the formation and management of high-level goals and the control of attention. Second, research on dopamine regulation has shown that sudden unpredicted visual/sound changes in the environment are detected by the superior colliculus, causing strong phasic bursts of dopamine. This has been proposed to have an important role in the formation of skills [Redgrave and Gurney, 2006, Mirolli et al., 2013].

Robots. Given their nature, IMs are ideally suited for driving progressive autonomous open-ended development in the absence of external rewards, tasks, and other types of external guidance. Computational analysis on IMs [Baldassarre and Mirolli, 2013] has resulted in systematisations of IM mechanisms relevant for exploiting them in robots. Oudeyer and Kaplan [2007] have distinguished between 'competence-based IMs (CB-IM), related to action (e.g., see Barto et al., 2004, Schembri et al., 2007) and 'knowledge-based IMs (KB-IMs)', related to sensorial aspects of stimuli. Successively, Baldassarre and Mirolli [2013] and Barto et al. [2013] have distinguished KB-IMs into 'novelty-based IMs (NB-IMs)' (e.g., see Nehmzow et al., 2013), related to the detection of novel stimuli not present in memory, and 'prediction-based IMs (PB-IMs)' (e.g., see Schmidhuber, 2010), related to the temporal anticipation of stimuli. In the past, IMs have been used as main means to guide the cumulative learning processes of autonomous robots (e.g., Schmidhuber, 1991, Oudeyer et al., 2007, Baldassarre and Mirolli, 2013). As we have argued in Santucci et al. [2012], however, IMs have usually been used to produce learning signals at each time step, e.g. in proportion to the size or change of a prediction error of a predictor predicting the effects of own actions. This had a very limited capacity to support the accumulation of skills.

Biological evidence mentioned above suggests a different use of IMs to guide open-ended learning. In particular, learning of skills needs to be organised around *few critical states of the world* becoming potential *goals* (or sub-goals) for future behaviour. IMs can play important roles when behaviour becomes organised around goals. First, NB-IMs can focus attention and exploration of the robot on novel aspects of the environment related to the goal. Second, CB-IMs can contribute to select those goals whose skills have a high rate of learning. Third, PB-IMs can play the critical role of supporting the *self-generation of goals* [Mirolli and Baldassarre, 2013]. For example, an accidental unpredicted effect of an agents own action can lead to the formation of the goal of causing that effect again and to improve the skill in doing so.

We have implemented models to test these ideas. In Baldassarre et al. [2013] and Fiore et al. [2014] NB-IMs drive bio-inspired embodied/robotic systems to focus attention on different parts of the environment where they have caused novel changes with own actions, and this allows them to acquire visual and manipulation skills. In Santucci et al. [2013] we used CB-IMs measuring skill improvement to select (externally assigned) goals where to focus learning resources. We are at the moment implementing and testing other architectures that extend these systems with the capacity of self-generating goals on the basis of PB-IMs (cf. Baranes and Oudeyer, 2013).

4 Concluding remarks

We have seen two ways in which looking at the brain gives suggestions about principles to follow to build open-ended learning robotic architectures. The first suggestion is to structure the overall architecture with three specific levels, each having its functioning and learning properties. The second suggestion is to use intrinsic motivations to self-generate critical goals and to drive and manage the acquisition of skills around them. There are other important indications from brain that we could not present here for lack of space. One with paramount importance concerns the need to endow the system with active vision [Hayhoe and Ballard, 2005] guided by *bottom-up and top-down attention* processes. This gives fundamental advantages to exploration and learning processes of autonomous robots [Marraffa et al., 2012, Ognibene and Baldassarre, 2014, Sperati and Baldassarre, 2014]. In conclusion, complex architectures are needed to build open-ended learning robots and those architectures can be built in innumerable, mostly wrong, ways. Thus, understanding how nature has structured those architectures could greatly help to meet the hard challenge of building truly open-ended learning robots.

References

G. Baldassarre. What are intrinsic motivations? A biological perspective. In *Proceedings of the International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob-2011)*, pages e1–8. IEEE, New York, NY, 2011.

G. Baldassarre and M. Mirolli, editors. *Intrinsically motivated learning in natural and artificial systems*. Springer, Berlin, 2013.

G. Baldassarre, Francesco M., V. G. Fiore, P. Redgrave, K. Gurney, and M. Mirolli. Intrinsically motivated action-outcome learning and goal-based action recall: A system-level bio-constrained computational model. *Neural Netw*, 41:168–187, 2013.

A. Baranes and P.Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robot Auton Syst*, 61(1):49–73, 2013.

A. Barto, M. Mirolli, and G. Baldassarre. Novelty or surprise? *Front Psychol*, 4(907):e1–15, 2013.

A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

A.G. Barto, S. Singh, and N. Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *International Conference on Developmental Learning (ICDL2004)*, New York, NY, 2004. IEEE.

D. E. Berlyne. Curiosity and exploration. *Science*, 143:25–33, 1966.

B. Castro da Silva, G. Baldassarre, G. Konidaris, and A. Barto. Learning parameterized motor skills on a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA2014)*, pages 1–6, 2014. Hong Kong, China.

V. G. Fiore, V Sperati, F. Mannella, M Mirolli, K. Gurney, K. Firston, R. J. Dolan, and G. Baldassarre. Keep focussing: striatal dopamine multiple functions resolved in a single mechanism tested in a simulated humanoid robot. *Frontiers in Psychology – Cognitive Science*, 5(124):e1–17, 2014.

V. Gallese and A. Goldman. Mirror neurons and the simulation theory of mind-reading. *Trends Cogn Sci*, 2(12):493–501, 1998.

M. S. A. Graziano. New insights into motor cortex. *Neuron*, 71(3):387–388, 2011.

S. Hart and R. Grupen. Learning generalizable control programs. *IEEE Trans Auton Mental Develop*, 3(1):216–231, 2011.

M. Hayhoe and D. Ballard. Eye movements in natural behavior. *Trends Cogn Sci*, 9(4):188–194, 2005.

A. J. Ijspeert and J. Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Artif Life*, 5(3):247–269, 1999.

A.J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 1523–1530. The MIT Press, Boston, MA, 2002.

J. Kober and J. Peters. Policy search for motor primitives in robotics. volume 21, pages 849–856. 2009.

G. D. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, pages 1468–1473, 2011.

D. Kumaran and E. A. Maguire. Which computational mechanisms operate in the hippocampus during novelty detection? *Hippocampus*, 17(9):735–748, 2007.

J. E. Lisman and A. A. Grace. The hippocampal-vta loop: controlling the entry of information into long-term memory. *Neuron*, 46(5):703–713, 2005.

R. Marraffa, V. Sperati, D. Caligiore, J. Triesch, and G. Baldassarre. A bio-inspired attention model of anticipation in gaze-contingency experiments with infants. In *IEEE International Conference on Development and Learning-EpiRob 2012 (ICDL-EpiRob-2012)*, pages e1–8, New York, NY, 2012. IEEE. San Diego, CA, 7-9 November 2012.

F. A. Middleton and P. L. Strick. Basal ganglia and cerebellar loops: motor and cognitive circuits. *Brain Res Brain Res Rev*, 31(2-3):236–250, Mar 2000.

E. K. Miller and J. D. Cohen. An integrative theory of prefrontal cortex function. *Annu Rev Neurosci*, 24:167–202, 2001.

M. Mirolli and G. Baldassarre. Functions and mechanisms of intrinsic motivations: the knowledge versus competence distinction. In G. Baldassarre and M. Mirolli, editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 49–72. Springer-Verlag, Berlin, 2013.

M. Mirolli, F. Mannella, and G. Baldassarre. The roles of the amygdala in the affective regulation of body, brain and behaviour. *Connect Sci*, 22(3):215–245, 2010.

M. Mirolli, V. G. Santucci, and G. Baldassarre. Phasic dopamine as a prediction error of intrinsic and extrinsic reinforcement driving both action acquisition and reward maximization: A simulated robotic study. *Neural Netw*, 39:40–51, 2013.

U. Nehmzow, E. Gatsoulis, Y. adn Kerr, J. Condell, N. Siddique, and T. M. McGinnity. Novelty detection as an intrinsic motivation for cumulative learning robots. In G. Baldassarre and M. Mirolli, editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 185–207. Springer-Verlag, Berlin, 2013.

D. Ognibene and G. Baldassarre. Ecological active vision: four bio-inspired principles to integrate bottom-up and adaptive top-down attention tested with a simple camera-arm robot. *IEEE Trans Auton Mental Develop*, 2014. doi: 10.1109/TAMD.2014.2341351.

P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? A typology of computational approaches. *Front Neurorobot*, 1:6, 2007.

P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Trans Evol Comp*, 11(2):265–286, 2007.

P. Redgrave and K. Gurney. The short-latency dopamine signal: a role in discovering novel actions? *Nature Rev Neurosci*, 7(12):967–975, 2006.

V. G. Santucci, G. Baldassarre, and M. Mirolli. Intrinsic motivation mechanisms for competence acquisition. In *Proceeding of the IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob 2012)*, pages e1–6. IEEE, 2012. San Diego, CA, 7-9 November 2012.

V. G. Santucci, G. Baldassarre, and M. Mirolli. Which is the best intrinsic motivation signal for learning multiple skills? *Front Neurorobot*, 7:e1–14, 2013. doi: 10.3389/fnbot.2013.00022.

V. G. Santucci, G. Baldassarre, and M. Mirolli. Autonomous selection of the "what" and the "how" of learning: an intrinsically motivated system tested with a two armed robot. In *The Fourth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob2014)*, pages e322–327, New York, NY, 2014. IEEE.

M. Schembri, M. Mirolli, and G. Baldassarre. Evolving childhood's length and learning parameters in an intrinsically motivated reinforcement learning robot. In Luc Berthouze, Prince G. Dhristiopher, Michael Littman, Hideki Kozima, and Christian Balkenius, editors, *Proceedings of the Seventh International Conference on Epigenetic Robotics*, volume 134, pages 141–148. Lund University, Lund, 2007.

J. Schmidhuber. Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1458–1463, 1991.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Trans Auton Mental Develop*, 2(3):230–247, 2010.

S. Singh, R.L. Lewis, A.G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Trans Auton Mental Develop*, 2(2):70–82, 2010.

V. Sperati and G. Baldassarre. Learning where to look with movement-based intrinsic motivations: a bio-inspired model. In *Proceedings of the Fourth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob2014)*, pages 453–460, New York, NY, 2014. IEEE. Genoa, Italy, 13–16 October.

F. Stulp, E. A. Theodorou, and S. Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Trans Robot*, 28(6):1360–2012, 2012.

R.S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif Int*, 112:181–211, 1999.

Decision Makers in a Changing Environment Anticipate Negative Changes and Resist Positive Changes.

Jason L. Harman
Social and Decision Sciences
Carnegie Mellon University
jharman@cmu.edu

Cleotilde Gonzalez
Social and Decision Sciences
Carnegie Mellon University
coty@cmu.edu

Abstract

Decision Makers in a Changing Environment Anticipate Negative Changes and Resist Positive Changes.

We examined decisions from experience with dynamic underlying probabilities. In a two-button choice task with a sure gain and a risky prospect between a high gain and no gain, we varied the probabilities of the risky option from .01 to1 over the course of 100 trials. Model simulations predict three phenomena: 1) When the high gain changes from certain to rare adaptation occurs rapidly, 2) when the high gain changes from rare to certain, adaptation occurs slowly, and 3) when held constant, choices drift towards the sure option. These predictions are confirmed by human behavior. One important deviation from human choice behavior is a much higher degree of lag when high gains change from rare to frequent.

Keywords: Risk, Decisions from Experience, Instance Based Learning Theory

Acknowledgements

This research is supported by the National Science Foundation Award number: 1154012 to Cleotilde Gonzalez. The authors would like to thank Hau-yu Wong for providing insightful comments.

1 Introduction

A common assumption in studies of decisions under risk is that risk is static and does not change over time. Though most paradigms reflect this assumption, many real world decisions are made in environments where risk is non-stationary. For example, most preventative health decisions are made to avoid a negative health events, but the probability of most negative health events increase over time making the decision environment different at any point in time. While changes in underlying probabilities may be known, such as increasing health risks, often information about changing risks is unknown or neglected. In these situations changes in risk may only be learned through feedback.

Rakow and Miler (2009) explored non-stationary risks in repeated decisions from feedback. Using a two button choice task common in studying experience based decision making (Barron & Erev, 2003), participants made repeated decisions between two options. At a given point in the game, the underlying probability of one option changed over ten trials making the expected value of that option either greater than or less than the expected value of the alternative option. Using multiple feedback manipulations over several stimuli pairs, Rakow and Miller found that participants choices did change with changing probabilities but were relatively slow to change, an observation they call the stickiness effect. They also replicated well known variance effect, where an option with less variability is preferred to an option with more variability[1] (Busemeyer & Towndsend, 1993). An intriguing effect reported by Rakow and Miller was that negative-to-positive changes elicited less stickiness behavior than positive-to-negative changes (direction-of-change effects). This effect however was not replicated in their second experiment, and an explanation of this observation was left for future work.

We attempt to provide an explanation for the direction-of-change effect in the current work, but also we extend the work by Rakow and Miler to rare outcomes where a very rare outcome becomes very likely and vice versa. In traditional decision making experiments, where participants are presented with fully described options in terms of possible outcomes and their associated probabilities, rare events are typically given more weight on a decision than they objectively deserve. In decisions based on experience however, the opposite has been found, with rare events having less impact of choice than they objectively deserve. Though there is some evidence that this finding may change over time (Harman & Gonzalez, in press), underweighting of rare outcomes in experience based decision making is a solid phenomenon regardless of sampling biases or the valence of the rare outcome. In the current work, the worst (best) outcome changes from rare to common over time. Model simulations predict that rare negative outcomes will have more influence on choice than rare positive outcomes.

1.1 A binary choice paradigm with changing probabilities

One option was always a sure gain of 250 points. The second option was a gamble that could result in either 500 points (high outcome) or 0 points (low outcome). In the first condition, the probability of obtaining the high outcome when the gamble is chosen begins at .01 and increases by .01 on each subsequent trial reaching a probability of 1 on the final trial. This condition represents an environment where rewards change from rare to certain over time. The second condition is a control condition where the probability of receiving the high outcome is .5 throughout all 100 trials (note that the expected value of the two options is equal on every trial in this condition). In the final condition, the probability of obtaining the high outcome when choosing the gamble is 1 at trial 1 and decreases by p = .01 on each subsequent trial until it is p = .01 at trial 100. This condition represents an environment where a reward changes from certain to rare over time.

2 Methods

2.1 Participants

240 participants (152 male, 88 female, mean age 31.32) completed the experiment on MTurk (U.S. IP restricted) and were compensated with both a set amount and an additional payment based on performance.

[1] given equal expected values and positive outcomes The opposite is true for negative outcomes (Weber et al. 2004)

2.2 Procedure

After providing consent and answering demographic questions, participants were given instructions for the game. Participants were paid $.50 for participating and an additional payment based on the amount of points accrued during the game (mean bonus payment = $ .26). Participants were randomly assigned to one of three conditions, each with a risky and dynamic probability option and a safe and stationary option (left/right presentation of options were counter-balanced).

Upon completion of the task, participants filled out a funnel debriefing designed to ascertain their knowledge of changes in the underlying probabilities. The funnel debriefing asked participants to describe the two options they choose from, whether they thought one was better than the other (gave the most points on average), and whether they thought one was better at one point in the game and worse at another point in the game. If a participant reported one option changed, they were asked whether the target option was better at the beginning or the end of the game and asked to indicate on a 0-100 trial sliding scale at which point they think the option became better (worse).

3 Modeling

We produced predictions via simulation using an Instance Based Learning (IBL) model of repeated binary choice (Gonzalez & Dutt, 2011). IBL is a decision making model that uses parts of the ACT-R cognitive architecture (Anderson & Libeaire, ***) to capture effects of memory (recency and frequency) in repeated choice. Simulation results are plotted in Figure 1 and discussed in the results section.

4 Results

We first collapse across trials to examine overall choice and performance. The three dependent variables we use for this analysis are the proportion of choices for the risky option (PRisky), the proportion of alternations from one option to another (Arate), and the total points earned in the task. The three conditions differed in the proportion of risky choices ($F(2, 296) = 30.03$, $p < .01$) with the highest proportion of risky choices in the decreasing condition (M=.48), followed by the static condition (M= .34) and the smallest proportion in the increasing condition (M=.27), all significantly different (LSD, $p < .01$). There was a significant difference in A rate between groups ($F(2, 296) = 10.62$, $p < .01$) driven by the increasing condition (M = .2) which was significantly lower than both the decreasing condition (M=.29) and the static condition (M= .29). There was also an effect in overall points earned ($F(2, 296) = 30.03$, $p < .01$). All three groups differed from each other with the highest number of points in the decreasing condition (M=27,388) followed by the increasing condition (M = 26,377) and the static condition (M= 24,862). To summarize the overall descriptive statistics using the static condition as the baseline, participants in the decreasing condition choose fewer risky options while participants in the increasing condition choose the risky option more frequently. This likely reflects early experience in the task which will be explored next. Alternation rates were lower in the increasing condition and were not different in the decreasing condition. Participants in both of the non-stationary conditions earned more points than those in the static condition.

Choice data for the three conditions over time is shown on the left hand side of figure 1. For each condition figure 1 plots the average choice proportion for the risky option on each trial. The dotted line in the two dynamic conditions plots the probability of obtaining the high (500) outcome when choosing the risky option. Because of the outcome structure, the dotted line can also be interpreted as the relative expected value of the risky option compared to the safe option on any given trial. For example at trial 50, the expected value of the risky option is 250 ([500 * p .5] + [0 * p .5]). The advantage of this representation is that the theoretical (dis)advantage of the risky option is apparent based on the distance (below) above the midpoint.

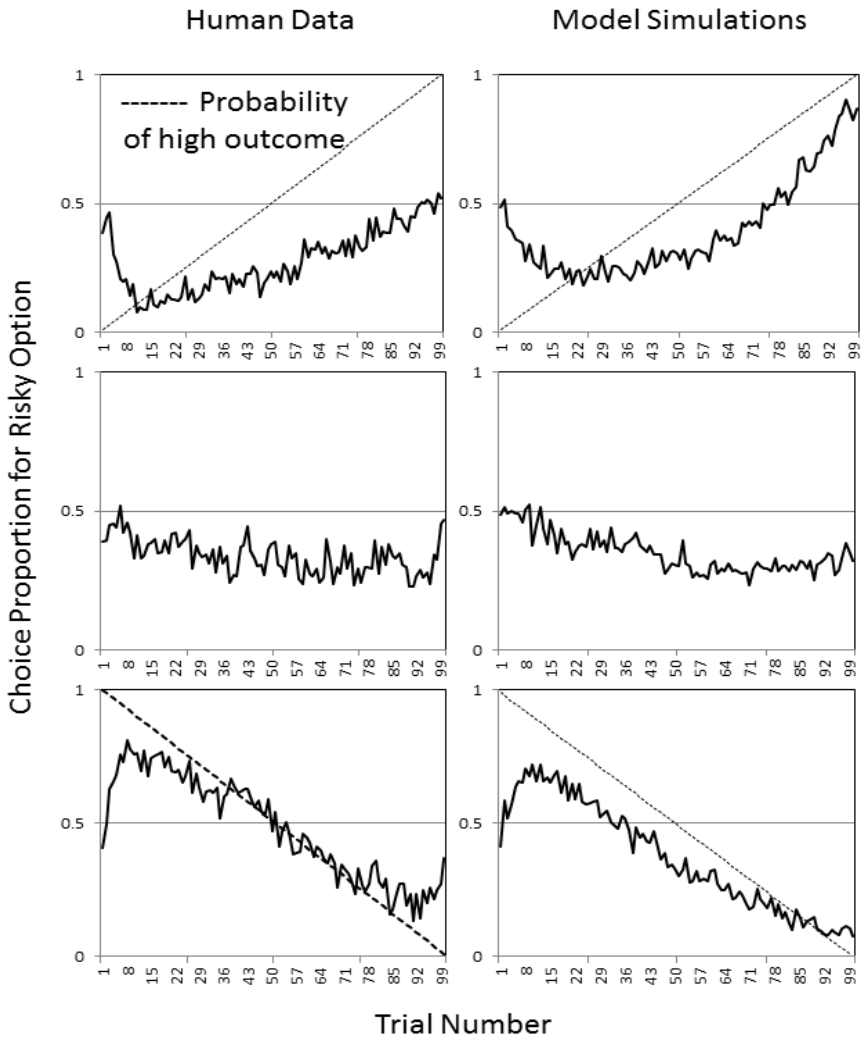


Figure 1. Choice proportion for the risky option over time for humans and the IBL Model.

Descriptively, the observed choice proportions are consistent with the patterns predicted from the IBL model. In the increasing probability condition (n = 76, top figures) where the probability of the high outcome goes from rare to certain, choice proportion quickly favors the safe option and slowly starts to abandon the safe option as trials progress; however humans on average only return to selecting the risky option only about half of the time as trials progress. In the stationary condition (n = 81), where the probability of the high outcome in the risky option is always .5 and the expected values of the two options are equal throughout the task, choice proportions favor the safe option over the course of the task. In the decreasing probability condition (n = 83, bottom figures), where the probability of the high outcome goes from certain to rare, choice proportions quickly favor the risky option and decrease as the probability of the high option decreases.

Two deviations from the model predictions are apparent. When the high outcome changes from rare to certain, the lag in choice proportions moving towards the risky option was more extreme than predicted, with choice proportions only reaching .5 by trial 100 (50 trials after the expected values favor the risky option). When the high outcome went from certain to rare, the change in choice proportion was slightly less extreme than predicted and choice proportions closely mirrored the underlying expected values (or the probability of the high outcome).

To provide a fit index, we calculated the MSD between the model predictions and the observed choice proportions. The

static condition had the best predictive fit with an MSD of 0.004466, followed by decreasing probability and increasing probability, with respective MSD's of 0.029 and 0.027847. In terms of optimal choice, the average number of choices from the option with the highest underlying expected value over the 100 trials was 66.93 in the decreasing condition and 58.45 in the increasing condition.

5 Discussion

Results from this study replicate Rakow and Milner's direction of change effect while extending it to more extreme changes in probability. Choice proportions were predicted well by the cognitive assumptions of IBL. One notable divergence between model simulations and observed data was the slow adaptation of humans in the negative to positive condition. Observed results suggest the possibility of a subset of participants that after experiencing negative outcomes under-sample as the game proceeds, never learning that the underlying probabilities have changed to make the risky option superior.

6 References

Barron, G., & Erev, I. (2003). Small feedback-based decisions and their limited correspondence to description-based decisions. *Journal Of Behavioral Decision Making, 16*(3), 215-233. doi:10.1002/bdm.443

Busemeyer, J. R., & Townsend, J. T. (1993). Decision field theory: A dynamic-cognitive approach to decision making in an uncertain environment. *Psychological Review, 100*(3), 432-459. doi:10.1037/0033-295X.100.3.432

Gonzalez, C., & Dutt, V. (2011). Instance-based learning: Integrating sampling and repeated decisions from experience. *Psychological Review, 118*(4), 523-551. doi:10.1037/a0024558

Harman, J. L., & Gonzalez, C. (*in press*). Allais from Experience: Choice consistency, rare events, and common consequences in repeated decisions. *Journal of Behavioral Decision Making.*

Rakow, T., & Miler, K. (2009). Doomed to repeat the successes of the past: History is best forgotten for repeated choices with nonstationary payoffs. *Memory & Cognition, 37*(7), 985-1000. doi:10.3758/MC.37.7.985

Approximate MaxEnt Inverse Optimal Control

De-An Huang
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA

Amir-massoud Farahmand*
Mitsubishi Electric Research Laboratories
Cambridge, MA, USA

Kris M. Kitani
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA

J. Andrew Bagnell
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA

Abstract

Maximum entropy inverse optimal control (MaxEnt IOC) is an effective means of discovering the underlying cost function of demonstrated agent's activity. To enable inference in large state spaces, we introduce an approximate MaxEnt IOC procedure to address the fundamental computational bottleneck stemming from calculating the partition function via dynamic programming. Approximate MaxEnt IOC is based on two components: approximate dynamic programming and Monte Carlo sampling. This approach has a finite-sample error upper bound guarantee on its excess loss. We validate the proposed method in the context of analyzing dual-agent interactions from video, where we use approximate MaxEnt IOC to simulate mental images of a single agents body pose sequence (a high-dimensional image space). We experiment with sequences image data taken from RGB data and show that it is possible to learn cost functions that lead to accurate predictions in high-dimensional problems that were previously intractable.[1]

Keywords: Inverse Optimal Control, Inverse Reinforcement Learning, Maximum Entropy Principle, Human Activity Recognition, Finite-Sample Error Bound

Acknowledgements

This research was sponsored in part by the Army Research Laboratory (W911NF-10-2-0061), the National Science Foundation (Purposeful Prediction: Co-robot Interaction via Understanding Intent and Goals), and the Natural Sciences and Engineering Research Council of Canada.

*This work was done while the author was at the Robotics Institute, Carnegie Mellon University.
[1]This work is a summary of Huang et al. [1].

1 Introduction

The Maximum Entropy (MaxEnt) Inverse Optimal Control (IOC) framework is an effective approach for discovering the underlying reward model of a rational agent [2, 3]. For instance, Kitani et al. [4] used MaxEnt IOC in the context of understanding and modeling human activities, where the recovered reward model encodes a person's set of preferences. They considered the computer vision problem of mentally (visually) simulate human activities. By integrating visual attributes of the scene as features of the reward function, they showed that highly accurate pedestrian trajectories could be simulated in novel scenes.

Most current approaches of MaxEnt IOC, however, are limited to problems with small state space. So for example, its application to visual prediction problems has been limited to 2D pedestrian trajectories. To extend MaxEnt IOC to deal with the inherent high-dimensional nature of observed human activity from image data, previous approaches [5, 6] relied on clustering techniques to quantize and reduce the size of the state space. However, coarse discretization of the state space resulted in non-smooth trajectories and inhibited the model's power to simulate the subtle qualities of activity dynamics. To address this problem, we recently introduced an *approximate MaxEnt IOC* algorithm that is suitable for dealing with problems with large or high-dimensional state space [1]. This paper summarizes the algorithm, the theoretical results, and the application of the framework in the context of analyzing dual-agent interactions from video.

At the heart of the problem of maximum entropy MaxEnt IOC and sequence prediction is an inference problem of computing the *log-partition function* that requires enumeration of all possible action sequences into the future given a set of observations. In the same way that the value function is computed for optimal control, the log-partition function of maximum entropy IOC can be computed using dynamic programming – differing only in the substitution of the "soft-max" operator for the "max" operator in the Bellman equations. This relationship was noted as early as [7] and formalized in [2]. While dynamic programming renders this efficient for small scale problems, more appropriate techniques are needed for dealing with problems with large state space.

When the state space is large, one natural approach is to use approximate dynamic programming for the approximate calculation of these functions. The approximate MaxEnt IOC algorithm of this paper in fact uses Approximate Value Iteration (AVI) to compute the softmax-based value (log-partition) function. The AVI procedure uses a regression estimator at each iteration. The choice of regression estimator is flexible and one can choose to work with local averagers, random forests, boosting, deep neural networks, etc. In this work, we particularly utilize a reproducing kernel Hilbert space-based (RKHS) regularized estimator due to its flexibility and favourable properties. We also briefly mention the theoretical properties of this algorithm and provide a finite-sample upper bound guarantee on the excess loss, i.e., the loss of our approximate procedure compared to an "ideal" MaxEnt IOC procedure without any approximation in the computation of the log-partition function or the feature expectation.

2 IOC for High-Dimensional Problems

The problem of the inverse optimal control, which is also known as inverse reinforcement learning, is to recover an agent's (or expert's) reward function based on its policy (or samples from the agent's behavior) when the dynamics of the process is known. Our approach to IOC is based on the Maximum Entropy Inverse Optimal Control of [3]. Let us first define a parametric-reward Markov Decision Process (θ-MDP). θ-MDP is defined as a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \underline{g}, \theta)$, where $\mathcal{X}$ is a state space, $\mathcal{A}$ is a finite set of actions, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$ is the transition probability kernel, $\underline{g} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^d$ is a mapping from state-action pairs to feature vectors of dimension d, and $\theta \in \mathbb{R}^d$ parametrizes the reward.[2] In our approach, the state space $\mathcal{X}$ can be large, e.g., $\mathbb{R}^D$. We consider θ-MDPs with finite horizon of T. Given a sequence $z_{1:T} = (z_1, \ldots, z_T)$, we denote $\underline{f}(z_{1:T}) = \sum_{t=1}^{T} \underline{g}(z_t)$. In IOC, we assume that $\mathcal{P}$ is known (or estimated separately).

Consider a set of demonstrated trajectories $\mathcal{D}_n = \{Z_{1:T}^{(i)}\}_{i=1}^n$ with each trajectory $Z_{1:T} = (Z_1, \ldots, Z_T) \sim \zeta$ with $Z_t = (X_t, A_t)$ and ζ being a distribution over the set of trajectory. Also denote $\nu \in \mathcal{M}(\mathcal{X})$ as the distribution of X_1. We only assume that the initial distribution ν is known, but the joint distribution ζ is not. For a policy π, denote $P_\pi(Z_{1:T})$ as the distribution induced by following policy π. In the discrete state case, $P_\pi(Z_{1:T}) = \prod_{t=1}^{T-1} \mathcal{P}(X_{t+1}|X_t, A_t)\pi(A_t|X_t)$ (and similarly for continuous state spaces). Define the *causal conditioned probability* $\mathbb{P}\{A_{1:T}||X_{1:T}\} = \prod_{t=1}^{T} \mathbb{P}\{A_t|X_t\} = \prod_{t=1}^{T} \pi_t(A_t|X_t)$, which reflects the fact that future states do not influence earlier actions (compare with conditional probability $\mathbb{P}\{A_{1:T}|X_{1:T}\}$). We define the *causal entropy* H_π as $H_\pi = \mathbb{E}_{P_\pi(Z_{1:T})}[-\log \mathbb{P}\{A_{1:T}||X_{1:T}\}]$.

The primal optimization problem in Maximum Entropy Inverse Optimal Control estimator [3] is

$$\arg\max_{\pi} H_\pi\left(A_{1:T}||X_{1:T}\right) \qquad \text{s.t.} \quad \mathbb{E}_{P_\pi(Z_{1:T})}\left[\underline{f}(Z_{1:T})\right] = \frac{1}{n}\sum_{i=1}^{n} \underline{f}\left(Z_{1:T}^{(i)}\right). \tag{1}$$

[2] $\mathcal{M}(\Omega)$ is the set of probability distributions over Ω.

Algorithm 1 – Backward pass

$\mathcal{D}_m^{(t)} = \{(X_i, A_i, R_i^t, X_i')\}_{i=1}^m$, $R_i^t = \left\langle \theta\,,\, \underline{g}(X_i, A_i) \right\rangle$
$\hat{Q}_T \leftarrow 0$
for $t = T-1, \ldots, 2, 1$ **do**
  $Y_i^t = R_i^t + \text{soft}\max \hat{Q}_{t+1}(X_i', \cdot)$
  $\hat{Q}_t \leftarrow \text{argmin}_Q \frac{1}{m} \sum_{i=1}^m \left| Q(X_i, A_i) - Y_i^t \right|^2 + \lambda_{Q,m} \|Q\|_{\mathcal{H}}^2$
  $\hat{\pi}_t(a|x) \propto \exp(\hat{Q}(x, a))$
end for

Algorithm 2 – Forward pass

$\underline{f} \leftarrow 0$
repeat
  $\hat{X}_1 \sim \nu$
  for $t = 1, \ldots T-1$ **do**
    $\hat{A}_t \sim \hat{\pi}_t(\cdot|\hat{X}_t)$, $\underline{f}$ += $\underline{g}^t(\hat{X}_t, \hat{A}_t)$
    $\hat{X}_{t+1} \sim \mathcal{P}(\cdot|\hat{X}_t, \hat{A}_t)$
  end for
until N sample paths
$\underline{f} \leftarrow \frac{1}{N} \underline{f}$ (estimated log-partition function gradient)

The motivation behind this objective function is to find a policy π whose induced expected features, $\mathbb{E}_{P_\pi(Z_{1:T})}\left[\underline{f}(Z_{1:T})\right]$, matches the empirical feature count of the agent, that is $\frac{1}{n}\sum_{i=1}^n \underline{f}(Z_{1:T}^{(i)})$, while not committing to any distribution beyond what is implied by the data. The dual of this constrained optimization problem is (Theorem 3 of [3])

$$\min_{\theta \in \mathbb{R}^d} \log \mathcal{Z}_\theta - \left\langle \theta\,,\, \frac{1}{n}\sum_{i=1}^n \underline{f}\left(Z_{1:T}^{(i)}\right) \right\rangle, \tag{2}$$

in which $\log \mathcal{Z}_\theta$ is the log-partition function (Theorem 3 of [3]). For notational compactness, define $\hat{b}_n, \bar{b} \in \mathbb{R}^d$ as $\hat{b}_n = \frac{1}{n}\sum_{i=1}^n \underline{f}(Z_{1:T}^{(i)})$ and $\bar{b} = \mathbb{E}_{Z_{1:T}\sim\zeta}\left[\underline{f}(Z_{1:T})\right]$. The vector $\bar{b}$ is the true expected feature of the agent, which is unknown.

A key observation is that one might calculate $\log \mathcal{Z}_\theta$ using a Value Iteration (VI) procedure: For any $\theta \in \mathbb{R}^d$, define $r_t(x, a) = r(x, a) = \left\langle \theta\,,\, \underline{g}(x, a) \right\rangle$, and perform the following VI procedure: Set $Q_T = r_T$, and for $t = T-1, \ldots, 1$,

$$Q_t(x, a) = r_t(x, a) + \int \mathcal{P}(dy|x, a) V_{t+1}(y), \qquad V_t(x) = \text{soft}\max(Q_t(x, \cdot) \triangleq \log\left(\sum_{a\in\mathcal{A}} \exp(Q_t(x, a))\right). \tag{3}$$

We compactly write $Q_t = r_t + \mathcal{P}^a V_{t+1}$, where $\mathcal{P}^a(\cdot|x) = \mathcal{P}(\cdot|x, a)$. It can be shown that $\log \mathcal{Z}_\theta = \mathbb{E}_\nu\left[V_1(X)\right]$. Also the MaxEnt policy solution to (1), which is in the form of Boltzmann distribution, is $\pi_t(a|x) = \pi_{t,\theta}(a|x) = \frac{\exp(Q_t(x,a))}{\sum_{a'\in\mathcal{A}} \exp(Q_t(x,a))} = \exp(Q_t(x, a) - V_t(x))$.

Instead of the original dual objective (2), we aim to solve the regularized dual objective

$$\min_{\theta \in \mathbb{R}^d} L(\theta, \hat{b}_n) \triangleq \log \mathcal{Z}_\theta - \left\langle \theta\,,\, \hat{b}_n \right\rangle + \frac{\lambda}{2}\|\theta\|_2^2, \tag{4}$$

which can be interpreted as a relaxation of the constraints in the primal [8]. It can be shown that $\nabla_\theta \log \mathcal{Z}_\theta = \mathbb{E}_{P_\pi(Z_{1:T})}\left[\underline{f}(Z_{1:T})\right]$ with $X_1 \sim \nu$, so the gradient of the loss function, which can be used in a gradient-descent-like procedure, is

$$\nabla_\theta L(\theta, \hat{b}_n) = \mathbb{E}_{P_\pi(Z_{1:T})}\left[\underline{f}(Z_{1:T})\right] - \hat{b}_n + \lambda\theta. \tag{5}$$

For problems with large state space, the exact calculation of the log-partition function $\log \mathcal{Z}_\theta$ is infeasible as is the calculation of the the expected features $\mathbb{E}_{P_\pi(Z_{1:T})}\left[\underline{f}(Z_{1:T})\right]$. Nonetheless, one can aim to approximate the log-partition function and estimate the expected features. We use two key insights to design an algorithm that can handle large state spaces. The first is that one can approximate the VI procedure of (3) using function approximators. The Approximate Value Iteration (AVI) procedure has been successfully used and theoretically analyzed in the Approximate Dynamic Programming and RL literature [9]. The second insight, which is also used in some previous work such as [10], is that one can estimate an expectation by Monte Carlo sampling and the error behavior would be $O(\frac{1}{\sqrt{N}})$ (for N independent trajectories), which is a dimension-free rate. These procedures are summarized in Algorithms 1 and 2. We describe each of them in detail.

To perform AVI, we use samples in the form of $\mathcal{D}_m^{(t)} = \{(X_i, A_i, R_i, X_i'\}_{i=1}^m$ with $X_i \sim \eta \in \mathcal{M}(\mathcal{X})$, $A_i \sim \pi_b(X_i)$, $R_i \sim \mathcal{R}(\cdot|X_i)$, and $X_i' \sim \mathcal{P}(\cdot|X_i, A_i)$. Here π_b is a behavior policy.[3] Given these samples, one can estimate Q_t with $\hat{Q}_t$ by

[3] In general, the distribution η used for the regression estimator is different from ζ. Furthermore, for simplicity of presentation we consider that η is fixed for all time steps, but this is not necessary. In practice one might choose to use $\mathcal{D}_m^{(t)} = \mathcal{D}_n^{(t)}$ extracted from the demonstrated trajectories $\mathcal{D}_n$.

solving a regression problem in which the input variables are $Z_i = (X_i, A_i)$ and the target values are $R_i + \hat{V}_{t+1}(X'_i)$, and $\hat{V}_{t+1} = \log\left(\sum_{a\in\mathcal{A}} \exp(\hat{Q}_t(x,a))\right)$. That is, $\hat{Q}_t \leftarrow \text{Regress}\left(\left\{\left((X_i, A_i), R_i + \hat{V}_{t+1}(X'_i)\right)\right\}_{i=1}^m\right)$.

Let us define $\tilde{Q}_t = r_t + \mathcal{P}^a \hat{V}_{t+1}$ and note that $\mathbb{E}\left[R_i + \hat{V}_{t+1}(X'_i)|(X_i, A_i)\right] = \tilde{Q}_t(X_i, A_i)$, i.e., $\tilde{Q}_t$ is the target regression function. We will shortly see that the quality of approximation, which is quantified by $\varepsilon_{\text{reg}}(t) \triangleq \|\hat{Q}_t - \tilde{Q}_t\|_2$, affects the excess error of approximate MaxEnt IOC procedure.

The choice of regression estimator is flexible in approximate MaxEnt IOC algorithm. It is desirable to use a powerful estimator that makes $\varepsilon_{\text{reg}}(t)$ as small as possible. One such a choice is the family of regularized least-squares estimators, which is also used in the Regularized Fitted Q-Iteration algorithm for control [11]: $\hat{Q}_t \leftarrow \text{argmin}_{Q\in\mathcal{F}^{|\mathcal{A}|}} \frac{1}{m}\sum_{i=1}^m \left|Q(X_i, A_i) - \left(R_i + \hat{V}_{t+1}(X'_i)\right)\right|^2 + \lambda_{Q,m} J(Q)$. Here $\mathcal{F}^{|\mathcal{A}|}$ is the set of action-value functions, $J(Q)$ is the regularization functional, which allows us to control the complexity, and $\lambda_{Q,m} > 0$ is the regularization coefficient. One particular choice is $\mathcal{F}^{|\mathcal{A}|}$ being a reproducing kernel Hilbert space (RKHS) and J being its corresponding norm, i.e., $J(Q) = \|Q\|^2_{\mathcal{H}}$. The AVI procedure with the RKHS-based formulation is summarized in Algorithm 1.

To estimate $\mathbb{E}_{P_\pi(Z_{1:T})}\left[f(Z_{1:T})\right]$ we may use Monte Carlo sampling: Draw a sample state from the initial distribution ν and then follow the sequence of policies π_t and count the features along the trajectory. Repeat this procedure N times (Algorithm 2).

Because of the approximation of AVI as well as the error caused by the Monte Carlo sampling, the solution $\tilde{\theta}_n$ of the approximate MaxEnt IOC procedure would have an error. We compare its loss to the ideal, but unavailable, case when the log-partition function could be solved exactly, the expectation was calculated exactly, and the true expected feature vector was available, i.e., $\min_{\theta\in\mathbb{R}^d} L(\theta, \bar{b})$. Huang et al. [1] provides a finite-sample error upper bound guarantee that compares the loss of our procedure, that is $L(\tilde{\theta}_n, \hat{b}_n)$, compared to the best possible loss assuming that the log-partition function could be solved exactly, the expectation was calculated exactly, and the true expected feature vector was available, i.e., $\min_{\theta\in\mathbb{R}^d} L(\theta, \bar{b})$. Under certain reasonable simplifying assumptions, the result is that for any $\delta > 0$, it holds that

$$L(\tilde{\theta}_n, \bar{b}) - \min_{\theta\in\mathbb{R}^d} L(\theta, \bar{b}) \le c\frac{T^3\sqrt{T\ln(1/\delta)}}{\lambda\sqrt{n}}\,\varepsilon_{\text{reg}},$$

with probability at least $1-\delta$.[4] Here ε_{reg} is an upper bound on the sequence $\left(\varepsilon_{\text{reg}}(t)\right)_{t=1}^{T-1}$. The value of $c > 0$ depends on the MDP and distributions ν and η through concentrability coefficients [12, 13, 14], and the number of actions. The regression error ε_{reg} depends on the regression estimator we use, the number of samples m, and the intrinsic difficulty of the regression problem characterized by its smoothness, sparsity, etc.

3 Mental Simulation of Human Interactions

We validate our approach in the context of analyzing dual-agent interactions from video, in which the actions of one person are used to predict the actions of another [5]. The key idea is that dual-agent interactions can be modelled as an optimal control problem, where the actions of the initiating agent induces a cost topology over the space of reactive poses – a space in which the reactive agent plans an optimal pose trajectory. Therefore, IOC can be applied to recover this underlying reactive cost function, which allows us to simulate mental images of the reactive body pose.

A visualization of the setting is shown in Figure 1a. As shown in the figure, the ground truth sequence contains both the true reaction sequence $q_{1:T} = (q_1, \ldots, q_T)$ on the left hand side (LHS) and the pose sequence of the initiating agent (obervation) $o_{1:T} = (o_1, \ldots, o_T)$ on the right hand side (RHS). At training time, n demonstrated interaction pairs $\{q^{(i)}_{1:T}\}_{i=1}^n$ and $\{o^{(i)}_{1:T}\}_{i=1}^n$ are provided to learn the reward model of human interaction. At test time, only the initiating actions on the RHS $o_{1:T}$ are observed, and we perform inference over the previously learned reactive model to obtain an optimal reaction sequence $x_{1:T}$. We follow [5] and model dual-agent interaction as an MDP in the following way. We use an 819-dimensional HOG feature [15] of an image patch around a person as our state (pose) representation. The actions are defined as the transition between states (poses), which are *deterministic* because we assume humans have perfect control over their body and one action will deterministically bring the pose to the next state.

Given two people interacting, we observe only the actions of the initiator on the RHS and attempt to simulate the reaction on the LHS. For evaluation, we used videos from *UT-interaction 1*, *UT-interaction 2* datasets [16]. The UTI datasets consist of RGB videos and has six actions: hand shaking, hugging, kicking, pointing, punching, pushing. In each interaction

[4] The simplifications are that $N \ge nT$, the number of samples m used in the regression estimation is in the same order as n, and the regression problem is not trivial, so ε_{reg} does not go to zero faster than $1/\sqrt{m}$. These are all reasonable assumptions.

(a)

AFD/NLL	NN	HMM	DMDP	KRL	Ours
shake	4.57/447	5.99/285	4.33/766	5.26/467	**4.08**/**213**
hug	4.78/507	8.89/339	**3.40**/690	4.11/475	3.53/**239**
kick	6.29/283	6.03/**184**	5.34/476	5.94/286	**3.92**/197
point	3.38/399	6.16/**321**	3.20/714	3.66/382	**3.06**/391
punch	3.81/246	5.85/193	4.06/396	4.71/254	**3.44**/**145**
push	4.21/315	7.73/214	**3.75**/446	4.67/324	3.94/**145**

(b)

Figure 1: (a) Examples of ground truth, partial observation, and visual simulation over occluded regions. (b) AFD and NLL per activity category for UTI

video, we occlude the ground truth reaction $q_{1:T} = (q_1, \ldots, q_T)$ on the LHS, observe $o_{1:T} = (o_1, \ldots, o_T)$ the action of the initiating agent on the RHS, and attempt to visually simulate $q_{1:T}$. Our baselines for comparisons are per frame nearest neighbor (NN), a hidden Markov model (HMM), discretized MDP-based (DMDP) MaxEnt IOC formulation [4], and the smoothing kernel-based RL (KRL) approach to MaxEnt IOC [5]. We compare the ground truth sequence with the learned policy using two metrics of Negative Log-Likelihood (NLL) $-\log P(q_{1:T}|o_{1:T}) = -\sum_t \log P(q_t|q_{t-1}, o_{1:T})$ and the average image feature distance (AFD) $\frac{1}{T}\sum_t ||q_t - x_t||^2$, where x_t is the resulting reaction pose at frame t.

The average NLL and image feature distance per activity for each baseline is shown in Figure 1b. To evaluate the accuracy of our Monte Carlo (MC) sampling algorithm, we compare with the Forward pass in [4] using our learned policy $\hat{\pi}$. Empirical results verify that our MC sampling strategy ($N = 500$) is able to achieve comparable performance. All optimal control based methods (DMDP and the proposed method) outperform the other two baselines in terms of image feature distance. Although the DMDP is able to achieve a lower than NN and HMM image feature distance, its NLL is worse then theirs. Furthermore, the proposed approximate MaxEnt IOC consistently outperforms the KRL value function approximation. Our method directly performs IOC on the continuous state space rather than interpolating value function of discretized state space. For more details, additional experiments, and comparison with other IOC and IRL algorithms refer to Huang et al. [1].

References

[1] De-An Huang, Amir-massoud Farahmand, Kris M Kitani, and J. Andrew Bagnell. Approximate MaxEnt inverse optimal control and its application for mental simulation of human interactions. In *AAAI*, January 2015.

[2] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

[3] Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. The principle of maximum causal entropy for estimating interacting processes. *IEEE Trans. on Information Theory*, 59(4):1966–1980, April 2013. ISSN 0018-9448.

[4] Kris M. Kitani, Brian D. Ziebart, J. Andrew Bagnell, and Martial Hebert. Activity forecasting. In *ECCV*, 2012.

[5] De-An Huang and Kris M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*, 2014.

[6] Jacob Walker, Abhinav Gupta, and Martial Hebert. Patch to the future: Unsupervised visual prediction. In *CVPR*, 2014.

[7] John Rust. Structural estimation of Markov decision processes. *Handbook of econometrics*, 4(4), 1994.

[8] Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. Performance guarantees for regularized maximum entropy density estimation. In *COLT*, volume 3120, pages 472–486. 2004.

[9] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *JMLR*, 9:815–857, 2008.

[10] Paul Vernaza and J Andrew Bagnell. Efficient high dimensional maximum entropy modeling via symmetric partition functions. In *NIPS*, pages 584–592, 2012.

[11] Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian Decision Problems. In *ACC*, pages 725–730, June 2009.

[12] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, pages 267–274, 2002.

[13] Rémi Munos. Performance bounds in L_p norm for approximate value iteration. *SIAM Journal on Control and Optimization*, pages 541–561, 2007.

[14] Amir-massoud Farahmand, Rémi Munos, and Csaba Szepesvári. Error propagation for approximate policy and value iteration. In *NIPS*. 2010.

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[16] Michael S. Ryoo and J. K. Aggarwal. UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA). http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html, 2010.

Automatic Generation of HTNs From PDDL

Anders Jonsson
University Pompeu Fabra
C/Roc Boronat 138
08018 Barcelona, Spain
anders.jonsson@upf.edu

Damir Lotinac
University Pompeu Fabra
C/Roc Boronat 138
08018 Barcelona, Spain
damir.lotinac@upf.edu

Abstract

Hierarchical Task Networks, or HTNs, are a popular model in planning for representing tasks or decision processes that are organized in a hierarchy. Although HTNs are known to be at least as expressive as STRIPS planning, being expressive enough to represent highly complex decision processes is not the main reason for their popularity. On the contrary, by imposing ordering constraints on the tasks at each level of the hierarchy, an HTN can significantly simplify the search for an action sequence that achieves a desired goal.

In this paper we present a novel algorithm that automatically generates HTNs from PDDL, the standard language for describing planning domains. The HTNs that our algorithm constructs contain two types of composite tasks that interact to achieve the goal of a planning instance. One type of task achieves fluents by traversing the edges of invariant graphs in which only one fluent can be true at a time. The other type of task traverses a single edge of an invariant graph by applying the associated action, which first involves ensuring that the preconditions of the action hold. The resulting HTNs can be applied to any instance of a planning domain, and are provably sound, such that the solution to an HTN instance can always be translated back to a solution to the original planning instance. In several domains we are able to solve most or all planning instances using HTNs created from a single example instance.

Keywords: planning, classical planning, hierarchical task networks

1 Introduction

Hierarchical Task Networks, or HTNs, are a popular tool for encoding expert knowledge in the form of a task hierarchy into a planning domain. Each task in the hierarchy has several possible decompositions, each involving a fixed set of subtasks and an associated partial order on subtasks. The solution to an HTN is a sequence of decompositions and ordering choices that results in a sequence of primitive tasks that is applicable in the initial state. HTNs have been successfully used in a variety of applications, usually by severely restricting the ordering choices to simplify the search for a solution. In the extreme case, each task only has one possible decomposition in which case no search at all is necessary to find a solution. Properly designing an HTN can be a time-consuming task for a human expert, but once this work is done, there is little need to optimize search.

We ask the following question: is it possible to devise an automatic, domain-independent approach for generating HTNs automatically from a basic description of a planning domain? Although there have been earlier attempts to generate HTNs automatically [4, 7], these approaches rely on partial information about the task decomposition. In contrast, we generate HTNs directly from the PDDL encoding of a planning domain and a single representative instance. Our approach is to generate HTNs that encode invariant graphs, which are similar to lifted domain transition graphs but can be subdivided on types. In experiments, we tested our approach on planning benchmarks from the International Planning Competition (IPC). In four domains, our algorithm is able to construct HTNs that make it possible to efficiently solve any instance using blind search. The approach is partially successful in other domains, but the branching factor becomes a problem for large instances. Still, the experimental results partially answers our question in the affirmative.

2 Planning Domains

We consider the fragment of PDDL that models typed STRIPS planning domains with positive preconditions and goals. A planning domain is a tuple $\boldsymbol{d} = \langle \mathcal{T}, \prec, P, A \rangle$, where $\mathcal{T}$ is a set of types, $\prec$ an inheritance relation on types, P a set of predicates and A a set of actions. Each predicate $p \in P$ and action $a \in A$ has a parameter list ($\varphi(p)$ and $\varphi(a)$, respectively) whose elements are types. Each action $a \in A$ has a precondition $\mathrm{pre}(a)$, an add effect $\mathrm{add}(a)$, and a delete effect $\mathrm{del}(a)$. Each precondition and effect consists of a predicate p and a mapping from $\varphi(p)$ to $\varphi(a)$.

Given a domain $\boldsymbol{d}$, a STRIPS planning instance is a tuple $\boldsymbol{p} = \langle \Omega, I, G \rangle$, where Ω is a set of objects, I is an initial state, and G is a goal state. Instance $\boldsymbol{p}$ implicitly defines a set F of propositional variables or *fluents* by assigning objects in Ω of the appropriate type to the parameters of each predicate, and a set O of *operators* by assigning objects in Ω to the parameters of each action. The initial state $I \subseteq F$ and goal state $G \subseteq F$ are both subsets of fluents.

Each operator $o \in O$ has a precondition $\mathrm{pre}(o) \subseteq F$, an add effect $\mathrm{add}(o) \subseteq F$ and a delete effect $\mathrm{del}(o) \subseteq F$, each a subset of fluents instantiated from the preconditions and effects of the associated action a. A state $s \subseteq F$ is a subset of fluents that are true, while fluents in $F \setminus s$ are false. An operator $o \in O$ is applicable in s if and only if $\mathrm{pre}(o) \subseteq s$, and the result of applying o in s is a new state $s \ltimes o = (s \setminus \mathrm{del}(o)) \cup \mathrm{add}(o)$. A plan for $\boldsymbol{p}$ is a sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ such that o_i, $1 \leq i \leq n$, is applicable in $I \ltimes o_1 \ltimes \cdots \ltimes o_{i-1}$, and π solves $\boldsymbol{p}$ if it reaches the goal state, i.e. if $G \subseteq I \ltimes o_1 \ltimes \cdots \ltimes o_n$.

3 Hierarchical Task Networks

We introduce a notation for HTN domains inspired by Geier and Bercher [2]. An HTN domain is a tuple $\boldsymbol{h} = \langle P, A, C, M \rangle$, where P is a set of predicates, A is a set of actions (i.e. primitive tasks), C is a set of compound tasks and M is a set of decomposition methods. Each task $c \in C$ and method $m \in M$ has an associated parameter list $\varphi(c)$ and $\varphi(m)$, respectively. Unlike STRIPS domains, HTN domains are untyped and we allow negative preconditions. A *task network* is a tuple $tn = \langle T, \prec \rangle$, where $T \subseteq A \cup C$ is a set of tasks and $\prec$ is a partial order on T. A method $m = \langle c, tn_m, \mathrm{pre}(m) \rangle$ consists of a compound task $c \in C$, a task network $tn_m = \langle T_m, \prec_m \rangle$ and a precondition $\mathrm{pre}(m)$. Each precondition $p \in P$ and task $t \in T_m$ has an associated mapping from $\varphi(p)$ or $\varphi(t)$ to $\varphi(m)$.

Given $\boldsymbol{h} = \langle P, A, C, M \rangle$, an HTN instance is a tuple $\boldsymbol{s} = \langle \Omega, I, tn_I \rangle$, where Ω is a set of objects, I is an initial state and tn_I is a task network. Just like for STRIPS, Ω induces sets F and O of fluents and operators, as well as sets $\mathcal{C}$ and $\mathcal{M}$ of grounded compound tasks and methods. A grounded task network has tasks in $O \cup \mathcal{C}$, and is *primitive* if all tasks are in O. The initial state $I \subseteq F$ is a subset of fluents, and the initial grounded task network $tn_I = \langle \{t_I\}, \emptyset \rangle$ has a single grounded compound task $t_I \in \mathcal{C}$.

We use $(s, tn) \rightarrow_D (s', tn')$ to denote that a pair of a state and a task network decomposes into another pair, where $tn = \langle T, \prec \rangle$ and $tn' = \langle T', \prec' \rangle$. A valid decomposition consists in choosing a task $t \in T$ such that $t' \not\prec t$ for each $t' \in T$, and applying one of the following rules:

1. If t is primitive, the decomposition is applicable if $\mathrm{pre}(t) \subseteq s$, and the resulting pair is given by $s' = s \ltimes t$, $T' = T \setminus \{t\}$ and $\prec' = \{(t_1, t_2) \in \prec \mid t_1, t_2 \in T'\}$.

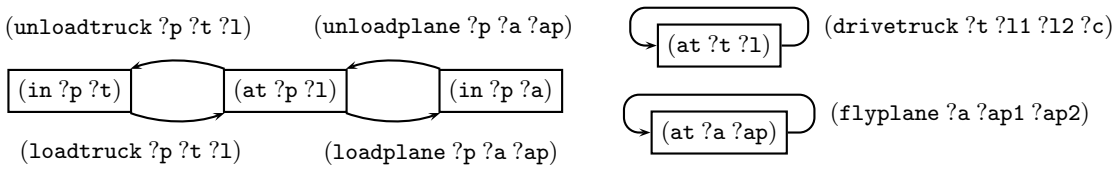


Figure 1: Invariant graphs in LOGISTICS.

2. If t is compound, the decomposition method $m = \langle t, tn_m, \mathrm{pre}(m)\rangle$ is applicable if $\mathrm{pre}(m) \subseteq s$, and the resulting pair is given by $s' = s$, $T' = T \setminus \{t\} \cup T_m$ and

$$\prec' = \{(t_1, t_2) \in \prec \mid t_1, t_2 \in T'\} \cup \{(t_1, t_2) \in T' \times T_m \mid (t_1, t) \in \prec\} \cup \{(t_2, t_1) \in T_m \times T' \mid (t, t_1) \in \prec\}.$$

The first rule removes a primitive task t from tn and applies the effects of t to the current state, while the second rule uses a method m to replace a compound task t with tn_m while leaving the state unchanged. If there is a finite sequence of decompositions from (s_1, tn_1) to (s_n, tn_n) we write $(s_1, tn_1) \to_D^* (s_n, tn_n)$. An HTN instance is solvable if and only if $(s_I, tn_I) \to_D^* (s_n, \langle\emptyset, \emptyset\rangle)$ for some state s_n, i.e. the resulting task network is empty.

4 Invariants

In STRIPS planning, a mutex invariant is a subset of fluents such that at most one is true at any moment. We generalize mutex invariants and use the Fast Downward planning system [3] to generate lifted (i.e. parameterized) invariants. We then use lifted invariants to construct *invariant graphs*. One reason our approach needs a representative instance is to test whether a lifted invariant corresponds to actual mutex invariants.

We illustrate the idea using the LOGISTICS domain. Fast Downward finds a single lifted invariant $\{$(in ?o ?v), (at ?o ?p)$\}$, i.e. a set of predicates with associated parameters. Parameters that appear across predicates (?o) are *bound* and take on the same value for all predicates. The remaining parameters (?v and ?p) are *free* and can be assigned any object of the appropriate type. Each assignment to bound parameters corresponds to a mutex invariant, formed by the set of fluents induced by all assignments to free parameters. The meaning of the lifted invariant for LOGISTICS is that across all instances, a given object ?o is either in a vehicle or at a location.

Given an invariant, our algorithm generates one or several invariant graphs by going through each action, finding each transition of each invariant that it induces (by pairing add and delete effects and testing whether the bound objects are identical), and mapping the types of the predicates to the invariant. We then either create a new invariant graph for the bound types or add nodes to an existing graph corresponding to the mapped predicate parameters.

Figure 1 shows the invariant graphs in LOGISTICS. In the top graph (G_1), the bound object is a package ?p, in the middle graph (G_2) a truck ?t, and in the bottom graph (G_3) an airplane ?a. Note that the predicate in is not actually part of the two bottom graphs, since trucks and planes cannot be inside other vehicles. Nevertheless, the invariant still applies: a truck or plane can only be at a single place at once. Each edge corresponds to an action that deletes one predicate of the invariant and adds another. To do so, the action has to include the parameters of both predicates, including the bound objects. In the figure, the invariant notation is extended to actions on edges, which have bound and free parameters.

5 Generating HTNs

In this section we describe our algorithm for automatically generating HTNs. The idea is to construct a hierarchy of tasks that traverse the invariant graphs to achieve certain fluents. In doing so there are two types of interleaved tasks: one that achieves a fluent in a given invariant (which involves applying a series of actions to traverse the edges of the graph), and one that applies the action on a given edge (which involves achieving the preconditions of the action).

Formally, our algorithm takes as input a STRIPS planning domain $\boldsymbol{d} = \langle \mathcal{T}, \prec, P, A\rangle$ and outputs an HTN domain $\boldsymbol{h} = \langle P', A', C, M\rangle$. The algorithm first constructs the invariant graphs $G_1, \ldots, G_k$ described above. Below we describe the components of the generated HTN domain $\boldsymbol{h}$.

5.1 Predicates and Tasks

The set $P' \supseteq P$ extends P with three predicates for each $p \in P$: persist-p, visited-p, and achieving-p. Respectively we use these predicates to temporarily cause p to persist, to flag p as an already visited node during search, and to prevent infinite recursion in case p or another predicate from the same invariant is currently being achieved.

(:method (achieve-p)
 ()
 ((set-flags-i) (achieve-p-i)
  (clear-flags-i) (lock-p)))

(:method (achieve-p-i)
 ((p') ($\neg$visited-p'))
 ((visit-p') (do-p'-a-i) (achieve-p-i)))

(:method (do-p-a-i)
 ()
 (((achieve-p_1) $\cdots$ (achieve-p_k)) (a)
  ((unlock-p_1) $\cdots$ (unlock-p_k))))

Figure 2: Outline of the generated methods.

Each action $a \in A$ from the input STRIPS planning domain $\boldsymbol{d}$ becomes a primitive task of $\boldsymbol{h}$. We add extra preconditions to ensure that a is not grounded on the wrong type, and that a does not delete a predicate that is supposed to persist. We also add primitive tasks for visiting, locking and unlocking each predicate $p \in P$. Visiting marks a predicate as visited, locking causes a predicate to temporarily persist, while unlocking frees a predicate so that it can be deleted again. For each invariant graph G_i, we add two primitive tasks set-flags-i that marks each predicate $p \in P$ in G_i as being achieved, and clear-flags-i that clears all flags for G_i.

We also include three types of compound tasks. For each predicate $p \in P$ that appears in any invariant graph, a task achieve-p that achieves p. For each invariant graph G_i and each $p \in P$ in G_i, a task achieve-p-i that achieves p in G_i. For each invariant graph G_i, each predicate $p \in P$ in G_i, and each outgoing edge of p (corresponding to an action $a \in A$), a task do-p-a-i. The first task is a wrapper task that achieves a predicate p in any invariant, while the other two are the interleaved tasks for achieving p by traversing the edges of an invariant graph G_i.

5.2 Methods

We describe the methods associated with each of the three types of compound tasks in turn. Since we use the HTN planner SHOP [6] to solve HTN instances, we outline each method in SHOP syntax in Figure 2.

The first type of task, achieve-p, has one associated method for each invariant graph G_i in which p appears. This method decomposes achieve-p into the task achieve-p-i, setting and clearing flags and locking p. The second type of compound task, achieve-p-i, has one associated method for each predicate p' in the invariant graph G_i and each outgoing edge of p' (corresponding to an action a that deletes p). Intuitively, one way to achieve p in G_i, given that we are currently at some different node p', is to traverse the edge associated with a using the compound task do-p'-a-i. Before doing so we mark p' as visited to prevent us from visiting p' again. After traversing the edge we recursively achieve p from the resulting node. To stop the recursion we define a "base case", a method that is applicable only when p holds and decomposes achieve-p-i into an empty task list.

The third type of compound task, do-p-a-i, has only one associated method that applies action a to traverse an outgoing edge of p in the invariant graph G_i. The tasks in the decomposition have to ensure that all preconditions $p_1, \ldots, p_k$ of a hold (excluding p, which holds by definition, as well as any static preconditions of a). Thus the method achieves all preconditions of a (locking them temporarily), then applies a, then unlocks the preconditions. To restrict the available choices when solving the HTN, we impose a total order on all tasks, except tasks (achieve-p_1) $\cdots$ (achieve-p_k) of the method do-p-a-i, since it may be difficult to determine in which order to achieve the preconditions of an action.

6 Optimizations

Achieving the preconditions of an action a in any order is inefficient since an algorithm solving the HTN instance may have to backtrack repeatedly. For this reason, we include an extension of our algorithm that uses a simple inference technique to compute a partial order in which to achieve the preconditions of a. We define a set of predicates whose value is supposed to persist, and check whether a path through an invariant graph is applicable given these persisting predicates. While doing so, only the values of bound variables are known, while free variables can take on any value. We match the bound variables of predicates and actions to determine whether an action allows a predicate to persist.

We use the same algorithm for precondition ordering to order a set of goal predicates P_G. We then order a set of fluents of each predicate $p \in P_G$ using a similar algorithm. To do so, the invariant graphs need to be partially grounded on each pair of fluents to be ordered. This partial grounding is done over the example instance only. The algorithm finds the indices of the parameters of p that invalidate the invariant, and generalize to any given instance of the planning domain.

7 Results

We ran our algorithm on the 9 typed STRIPS planning domains from IPC-2000 and IPC-2002, in order to directly compare the performance of our automatically generated HTNs with hand-crafted HTNs. Since hand-crafted planners were not allowed to compete in later competitions, there exist no hand-coded HTNs from those competitions to compare to.

	HTNPrecon			HTNGoal			FDBlind			Hand-crafted		
FREECELL[60]	0	-	-	0	-	-	5	228	17834	60	-	-
BLOCKS[35]	0	-	-	12	50.84	6877	18	32.5	7856	35	0.3	0
ROVERS[20]	20	1.7	16	20	2.0	16	6	219	32787	20	1	1
LOGISTICS[80]	80	8.3	75	80	29.2	75	10	1.58	432	80	-	-
DRIVERLOG[20]	7	74.5	5080	8	60.1	4913	7	40.2	3421	20	-	-
ZENOTRAVEL[20]	4	1527.8	194477	6	1453.8	161365	8	162	5994	20	0.2	0
MICONIC[150]	150	0.66	0	150	0.75	0	55	509	75372	150	0.0	0
SATELLITE[20]	18	0.59	1.2	20	1	0.7	6	702	22982	20	-	-
DEPOTS[22]	4	59.73	4655	15	1178.8	50404	4	53.5	6034	22	-	-

Table 1: Results in the IPC-2000 and IPC-2002 domains, with the number of instances of each domain shown in brackets.

We performed experiments with two versions of our algorithm. The base algorithm that achieves the preconditions and goals in any order was slow in testing, so we activated precondition ordering in both versions. The first version, HTNPrecon, achieves the goals in the order they appear in the PDDL definition. The second version, HTNGoal, implements our goal ordering strategy in addition to precondition ordering.

Table 1 shows the results for the 9 domains. For each planner we report the number of instances solved and the maximum time taken to solve an instance. We also report the maximum number of backtracks (in thousands). For hand-crafted domains which could not be solved by the JSHOP planner we provide only coverage. As expected, our goal ordering strategy mainly improves the performance of the algorithm in BLOCKS and DEPOTS, the two domains that are most sensitive to goal ordering. In addition, goal ordering enables us to solve two additional instances in SATELLITE. The hand-crafted HTNs successfully solve all instances of all domains with little backtracking; however, our algorithm generates HTNs in a fraction of a second, while the hand-crafted HTNs were carefully designed by human experts.

8 Conclusion

In this paper we have presented what we believe to be the first domain-independent algorithm for generating HTNs. All the algorithm needs is a PDDL description of the planning domain and a single representative instance. In four domains, the algorithm successfully generates HTNs that can be used to efficiently solve any instance, thus being competitive with HTNs designed by human experts and heuristic search algorithms. Although the success of the algorithm is limited in the remaining domains, we believe that there are still many potential benefits. In many cases, even though the resulting HTN is not constrained enough, subtasks identified by the algorithm may still be useful. This is the case, for example, in BLOCKS, where the resulting HTN contains tasks and methods for putting a block on top of another block. In such cases, the algorithm can help suggest initial decompositions that can later be refined by a human expert.

The avenue for future research that we find most promising is to test different restrictions on the invariant graphs. If the representative instance can still be solved under some restriction, the resulting HTN may still be able to solve other instances, and the restriction has the effect of reducing the branching factor. In essence, this mechanism would reduce the number of ways to traverse the invariant graphs.

Another option is to translate the resulting HTNs back to classical planning instead of using an HTN solver [1, 5]. In this way we can take advantage of the reduced branching factor offered by the HTNs, and use heuristic search planners to solve the resulting classical planning instances.

References

[1] R. Alford, U. Kuter, and D. Nau. Translating HTNs to PDDL: A Small Amount of Domain Knowledge Can Go a Long Way. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1629–1634, 2009.

[2] T. Geier and P. Bercher. On the Decidability of HTN Planning with Task Insertion. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 1955–1961, 2011.

[3] M. Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

[4] C. Hogg, H. Munoz-Avila, and U. Kuter. HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, pages 950–956, 2008.

[5] M. Lekavý and P. Návrat. Expressivity of STRIPS-Like and HTN-Like Planning. In *Proceedings of the 1st KES Symposium on Agent and Multi-Agent Systems - Technologies and Applications (KES-AMSTA'07)*, pages 121–130, 2007.

[6] D. Nau, T. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[7] H. Zhuo, D. Hu, C. Hogg, Q. Yang, and H. Munoz-Avila. Learning HTN Method Preconditions and Action Models from Partial Observations. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1804–1809, 2009.

Reinforcement Learning in Multi-Agent Systems with Partial History Sharing

Jalal Arabneydi
Department of Electrical and Computer Engineering
McGill University
Montreal, QC H3A 0E9
jalal.arabneydi@mail.mcgill.ca

Aditya Mahajan
Department of Electrical and Computer Engineering
McGill University
Montreal, QC H3A 0E9
aditya.mahajan@mcgill.ca

Abstract

In this paper, we are interested in systems with multiple agents that wish to cooperate in order to accomplish a common task while a) agents have different information (decentralized information) and b) agents do not know the complete model of the system i.e., they may only know the partial model or may not know the model at all. The agents must learn the optimal strategies by interacting with their environment i.e., by multi-agent Reinforcement Learning (RL). The presence of multiple agents with different information makes multi-agent (decentralized) reinforcement learning conceptually more difficult than single-agent (centralized) reinforcement learning.

We propose a novel multi-agent reinforcement learning algorithm that learns ϵ-team-optimal solution for systems with partial history sharing information structure, which encompasses a large class of multi-agent systems including delayed sharing, control sharing, mean field sharing, etc. Our approach consists of two main steps as follows: 1) the multi-agent (decentralized) system is converted to an equivalent single-agent (centralized) POMDP (Partial Observable Markov Decision Process) using the common information approach of Nayyar et al, TAC 2013, and 2) based on the obtained POMDP, an approximate RL algorithm is constructed using a novel methodology. Particularly, in the second step, since the POMDP obtained in the first step requires the complete-knowledge of system model, we introduce a new concept that we call "Incrementally Expanding Representation (IER)". The main feature of IER is to remove the dependency of the POMDP from complete-knowledge of the model. Then, based on an appropriately defined IER, we follow three sub-steps: 2a) convert the POMDP to a countable-state MDP Δ, 2b) approximate Δ with a sequence of finite-state MDPs $\{\Delta_N\}_{N=1}^{\infty}$, and 2c) use a RL algorithm to learn optimal strategy of MDP Δ_N.

We show that the performance of the RL strategy converges to the optimal performance exponentially fast. We illustrate the proposed approach and verify it numerically by obtaining a multi-agent Q-learning algorithm for two-user Multi Access Broadcast Channel (MABC) which is a benchmark example for multi-agent systems.

Keywords: Reinforcement Learning, Multi-agent Systems, Stochastic Systems.

Acknowledgements

We are deeply indebted to iCORE for their generous support of RLDM2015.

1 Introduction

In this paper, we propose a multi-agent Reinforcement Learning (RL) algorithm that guarantees team-optimal solution. Existing approaches for multi-agent learning may be categorized as follows: exact methods and heuristics. The exact methods rely on the assumption that the information structure is such that all controllers can consistently update the Q-function. These include approaches that rely on social convention and rules to restrict the decisions made by the controllers [5]; approaches that use communication to convey the decisions to all controllers [6]; and approaches that assume that the Q-function decomposes into a sum of terms, each of which is independently updated by a controller [7]. Heuristic approaches include joint action learners heuristic [8] where each controller learns the empirical model of the system in order to estimate the control action of other controllers; frequency maximum Q-value heuristic [9] where controllers keep track of the frequency with which each action leads to a "good" outcome; heuristic Q-learning [10] which assigns a rate of punishment for each controller; and distributed Q-learning [11] which uses predator-prey models to assign heuristic sub-goals to individual controllers. To best of our knowledge, there is no RL approach that guarantees team optimal solution. In this paper, we present an approach that guarantees team-optimal solution.

2 System Model

Let $X_t \in \mathcal{X}$ denote the state of a dynamical system controlled by n agents. At time t, agent i observes $Y_t^i \in \mathcal{Y}^i$ and chooses $U_t^i \in \mathcal{U}^i$. For ease of notation, we denote the joint action and the joint observation by $\mathbf{U}_t = (U_t^1, \dots, U_t^n)$ and $\mathbf{Y}_t = (Y_t^1, \dots, Y_t^n)$, respectively. The dynamics of the system are given by

$$X_{t+1} = f(X_t, \mathbf{U}_t, W_t^s), \tag{1}$$

and the observations are given by

$$\mathbf{Y}_t = h(X_t, \mathbf{U}_{t-1}, W_t^o). \tag{2}$$

In this paper, all system variables are considered finite valued. Let $I_t^i \subseteq \{\mathbf{Y}_{1:t}, \mathbf{U}_{1:t-1}\}$ be information available at agent i at time t. The collection $(\{I_t^i\}_{t=1}^{\infty}, i = 1, \dots, n)$ is called the *information structure*. In this paper, we restrict attention to an information structure called *partial history sharing* (PHS) [1], which will be defined later.

At time t, agent i chooses action U_t^i according to *control law* g_t^i as follows

$$U_t^i = g_t^i(I_t^i). \tag{3}$$

We denote $\mathbf{g}^i = (g_1^i, g_2^i, \dots)$ as *strategy* of agent i and $\mathbf{g} = (\mathbf{g}^1, \dots, \mathbf{g}^n)$ as joint strategy of agents. The performance of strategy $\mathbf{g}$ is measured by the following infinite-horizon discounted cost

$$J(\mathbf{g}) = \mathbb{E}^{\mathbf{g}} \left[\sum_{t=1}^{\infty} \beta^{t-1} \ell(\mathbf{X}_t, \mathbf{U}_t) \right], \tag{4}$$

where discount factor $\beta \in (0, 1)$. We are interested in the following problem.

Problem 1 *Given the information structure, action spaces $\{\mathcal{U}^i\}_{i=1}^n$, observation spaces $\{\mathcal{Y}^i\}_{i=1}^n$, discount factor β, and any $\epsilon > 0$, develop a (model-based or model-free) reinforcement learning algorithm that guarantees an ϵ-optimal strategy $\mathbf{g}^*$.*

3 Preliminaries on Partial History Sharing

Herein, we present a simplified version of partial history sharing information structure, originally presented in [1].

Definition 1 ([1], Partial History Sharing (PHS)) *Consider a decentralized control system with n agents. Let I_t^i denote the information available to agent i at time t. Assume $I_t^i \subseteq I_{t+1}^i$. Then, split the information at each agent into two parts: common information $C_t = \bigcap_{i=1}^n I_t^i$ i.e. the information shared between all agents and local information $M_t^i = I_t^i \backslash C_t$ that is the local information of agent i. Define $Z_t := C_{t+1} \backslash C_t$ as common observation, then $C_{t+1} = Z_{1:t}$. An information structure is called partial history sharing when the following conditions are satisfied:*

- *a) The update of local information $M_{t+1}^i \subseteq \{M_t^i, U_t^i, Y_{t+1}^i\} \backslash Z_t, \quad i \in \{1, \dots, n\}$.*
- *b) For every agent i, the size of local information M_t^i and the size of common observation Z_t are uniformly bounded in time t.*

These conditions are fairly mild and are satisfied by a large class of models.

Remark 1 *Note that conditions (a) and (b) are valid even if there is no common information between agents i.e., $C_t = \emptyset$.*

4 Approach

In this section, we derive our results for systems that have partial history sharing information structure defined above. Our approach consists of two steps. In the first step, we consider the setup of the complete-knowledge of the model and use the *common information approach* of [1] to convert the multi-agent system with PHS information structure to an equivalent single-agent POMDP. In the second step, based on the obtained POMDP, we develop an approximate RL algorithm for the setup of incomplete-knowledge of the model. We show that the error associated with the approximate RL converges to zero exponentially fast.

4.1 Step 1: An Equivalent single-agent POMDP

In this section, we present common information approach of [1] and its main results for the setup of complete-knowledge of the model described in Section 2. Let $\Gamma_t^i : \mathcal{M}^i \mapsto \mathcal{U}^i$ be the mapping from the local information of subsystem i to action of subsystem i at time t i.e. $U_t^i = \Gamma_t^i(M_t^i)$.

Consider a virtual *coordinator* (single agent) that observes the common information shared between all subsystems by time t i.e. C_t. Based on C_t, the coordinator prescribes functions $\mathbf{\Gamma}_t = (\Gamma_t^1, \dots, \Gamma_t^n) \in \mathcal{G}$ to subsystems, where $\mathcal{G} = \prod_{i=1}^n \mathcal{G}^i$ denotes the space of joint mappings $\mathbf{\Gamma}_t$ and $\mathcal{G}^i$ denotes the space of mappings Γ_t^i. Hence, $\Gamma_t^i = \psi_t^i(C_t), \quad \forall i \in \{1, \dots, n\}$ where $\boldsymbol{\psi}_t = \{\psi_t^1, \dots, \psi_t^n\}$ is called the *coordination law* and $\mathbf{\Gamma}_t = (\Gamma_t^1, \dots, \Gamma_t^n)$ is called the *prescription*. In the sequel, for ease of notation, we will use the following compact form for the coordinator's law, $\mathbf{\Gamma}_t = \boldsymbol{\psi}_t(C_t)$. We call $\boldsymbol{\psi} = \{\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \dots\}$ as the *coordination strategy*. In the *coordinated system*, dynamics and cost function are as same as those in the original problem in Section 2. In particular, the infintie-horizon discounted cost in the coordinated system is as follows:

$$J(\boldsymbol{\psi}) = \mathbb{E}^{\boldsymbol{\psi}} \left[\sum_{t=1}^{\infty} \beta^{t-1} \ell(\mathbf{X}_t, \mathbf{\Gamma}_t^1(M_t^1), \dots, \Gamma_t^n(M_t^n)) \right]. \tag{5}$$

Lemma 1 ([1], Proposition 3) *The original system described in Section 2 with PHS information structure is equivalent to the coordinated system.*

According to [1], $\Pi_t = \mathbb{P}(\mathbf{X}_t, \mathbf{M}_t | Z_{1:t-1}, \mathbf{\Gamma}_{1:t-1})$ is an information state for the coordinated system with initial state $\Pi_1 = P_X$. It is shown in [1] that

1. There exists a function ϕ such that $\Pi_{t+1} = \phi(\Pi_t, \mathbf{\Gamma}_t, Z_t)$.
2. The observation Z_t only depends on (Π_t, Γ_t) i.e. $\mathbb{P}(Z_t{=}z_t | \Pi_{1:t}{=}\pi_{1:t}, \mathbf{\Gamma}_{1:t}{=}\boldsymbol{\gamma}_{1:t}) = \mathbb{P}(Z_t{=}z_t | \Pi_t{=}\pi_t, \mathbf{\Gamma}_t{=}\boldsymbol{\gamma}_t)$.
3. There exists a function $\hat{\ell}$ such that $\hat{\ell}(\pi_t, \boldsymbol{\gamma}_t) = \mathbb{E}[\ell(\mathbf{X}_t, \mathbf{U}_t | Z_{1:t-1}{=}z_{1:t-1}, \mathbf{\Gamma}_{1:t}{=}\boldsymbol{\gamma}_{1:t})]$.

Assume that the initial state π_1 is fixed. Let $\mathcal{R}$ denote the reachable set of above centralized POMDP that contains all the realizations of π_t generated by $\pi_{t+1} = \phi(\pi_t, \boldsymbol{\gamma}, z), \forall \boldsymbol{\gamma} \in \mathcal{G}, \forall z \in \mathcal{Z}, \forall t \in \mathbb{N}$, with initial information state π_1. Note that since all the variables are finite valued, then $\mathcal{G}$ (set of all prescriptions $\boldsymbol{\gamma}$) and $\mathcal{Z}$ (set of all observations of the coordinator) are finite sets. Hence, $\mathcal{R}$ is at most a countable set. In the next step, we develop an approximate RL algorithm based on the obtained POMDP for the setup of incomplete-knowledge of the model.

4.2 Step 2: An Approximate RL algorithm for POMDP

In the previous step, we identified a single-agent POMDP that is equivalent to the multi-agent system with PHS information structure. However, the obtained POMDP requires the complete knowledge of the model. To circumvent this requirement, we introduce a new concept that we call *Incrementally Expanding Representation* (IER). The main feature of IER is to remove the dependency of the POMDP from the complete knowledge of the model. Using the IER, we follow three sub-steps: 2a) convert the POMDP to a countable-state MDP Δ, 2b) construct a sequence of finite-state MDPs $\{\Delta_N\}_{N=1}^{\infty}$ of MDP Δ, and 2c) use a generic RL algorithm to learn an optimal strategy of Δ_N.

Definition 2 (Incrementally Expanding Representation (IER)) *Let $\{\mathcal{S}_k\}_{k=1}^{\infty}$ be a sequence of finite sets such that $\mathcal{S}_1 \subsetneq \mathcal{S}_2 \subsetneq \dots \subsetneq \mathcal{S}_k \subsetneq \dots$, and $\mathcal{S}_1$ is a singleton, say $\mathcal{S}_1 = \{s^*\}$. Let $\mathcal{S} = \lim_{k\to\infty} \mathcal{S}_k$ be the countable union of above finite sets, $B : \mathcal{S} \to \mathcal{R}$ be a sujrjective function that maps $\mathcal{S}$ to the reachable set $\mathcal{R}$, and $\tilde{f} : \mathcal{S} \times \mathcal{G} \times \mathcal{Z} \to \mathcal{S}$. The tuple $\langle \{\mathcal{S}_k\}_{k=1}^{\infty}, B, \tilde{f} \rangle$ is called an* incrementally expanding representation *(IER), if it satisfies the following properties:*

(P1) Incremental Expansion: For any $\boldsymbol{\gamma} \in \mathcal{G}, z \in \mathcal{Z}$, and $s \in \mathcal{S}_k$, we have that

$$\tilde{f}(s, \boldsymbol{\gamma}, z) \in \mathcal{S}_{k+1}. \tag{6}$$

(P2) Consistency: For any $(\boldsymbol{\gamma}_{1:t-1}, z_{1:t-1})$, let Π_t be the information state of the obtained POMDP and S_t be the state obtained by recursive application of (6) *starting from $S_1 = s^*$. Then, $\Pi_t = B(S_t)$.*

Lemma 2 *Every multi-agent system with PHS information structure has at least one IER.*

4.2.1 Countable-state MDP Δ

Let the tuple $\langle \{\mathcal{S}_k\}_{k=1}^{\infty}, B, \tilde{f} \rangle$ be an IER of the POMDP obtained in the first step. Then, define MDP Δ with countable state space $\mathcal{S}$, finite action space $\mathcal{G}$, and dynamics $\tilde{f}$ such that:

(F1) $\mathcal{S} = \lim_{k\to\infty} S_k$ is the (countable) state space and $\mathcal{G}$ is the finite action space of MDP Δ. The initial state is singleton s^*. The state $S_t \in \mathcal{S}_k$, $k \leq t$, evolves as follows:

$$S_{t+1} = \tilde{f}(S_t, \mathbf{\Gamma}_t, Z_t), \quad S_{t+1} \in \mathcal{S}_{k+1}, \mathbf{\Gamma}_t \in \mathcal{G}, Z_t \in \mathcal{Z},$$

where observation Z_t only depends on $(S_t, \mathbf{\Gamma}_t)$. At time t, there is a cost depending on the current state $S_t \in \mathcal{S}$ and action $\mathbf{\Gamma}_t \in \mathcal{G}$ given by $\tilde{\ell}(S_t, \mathbf{\Gamma}_t) = \hat{\ell}(B(S_t), \mathbf{\Gamma}_t) = \hat{\ell}(\Pi_t, \mathbf{\Gamma}_t)$.

(F2) State space $\mathcal{S}$, action space $\mathcal{G}$, and dynamics $\tilde{f}$ do not depend on the unknowns.

The performance of a stationary strategy $\tilde{\boldsymbol{\psi}} : \mathcal{S} \mapsto \mathcal{G}$ is quantified by $\tilde{J}(\tilde{\boldsymbol{\psi}}) = \mathbb{E}^{\tilde{\boldsymbol{\psi}}} \left[\sum_{t=1}^{\infty} \beta^{t-1} \tilde{\ell}(S_t, \mathbf{\Gamma}_t) \right]$.

Lemma 3 *There exists at least one Δ that satisfies F1 and F2. Also, let $\tilde{\boldsymbol{\psi}}^*$ be an optimal strategy of MDP Δ. Construct a strategy $\boldsymbol{\psi}^*$ for the coordinated system as follows: $\tilde{\boldsymbol{\psi}}^*(s) =: \boldsymbol{\psi}^*(B(s)), \forall s \in \mathcal{S}$. Then, $\tilde{J}(\tilde{\boldsymbol{\psi}}^*) = J(\boldsymbol{\psi}^*)$ and $\boldsymbol{\psi}^*$ is an optimal strategy for the coordinated system, and therefore can be used to generate an optimal strategy for the original multi-agent system.*

4.2.2 Finite-state incrementally expanding MDP Δ_N

In this part, we construct a series of finite-state MDPs $\{\Delta_N\}_{N=1}^{\infty}$, that approximate the countable-state MDP Δ as follows. Let Δ_N be a finite-state MDP with state space $\mathcal{S}_N$ and action space $\mathcal{G}$. The transition probability of Δ_N is constructed as follows. Pick any arbitrary set $D^* \in S_N$. Remap every transition in Δ that takes the state $s \in \mathcal{S}_N$ to $s' \in \mathcal{S}_{N+1} \backslash \mathcal{S}_N$ to a transition from $s \in \mathcal{S}_N$ to any (not necessarily unique) state in D^*. In addition, the per-step cost function of Δ_N is simply a restriction of $\tilde{\ell}$ to $\mathcal{S}_N \times \mathcal{G}$. Also, we assume that there exists an action or a sequence of actions that if taken, the system transmits to a known state (states) d^* in D^*. Then, dynamics of Δ_N is as follows.

$$S_{t+1} = \begin{cases} \tilde{f}(S_t, \mathbf{\Gamma}_t, Z_t) & \tilde{f}(S_t, \mathbf{\Gamma}_t, Z_t) \in \mathcal{S}_N \\ d^* & \tilde{f}(S_t, \mathbf{\Gamma}_t, Z_t) \in \mathcal{S}_{N+1} \backslash \mathcal{S}_N \end{cases} \tag{7}$$

Theorem 1 *Let $\tilde{\boldsymbol{\psi}}^*$ be an optimal strategy of MDP Δ and $\tilde{\boldsymbol{\psi}}_N^*$ be an optimal strategy of MDP Δ_N. Then, the difference in performance is bounded as follows: $|\tilde{J}(\tilde{\boldsymbol{\psi}}^*) - \tilde{J}_N(\tilde{\boldsymbol{\psi}}_N^*)| \leq \frac{2\beta^{\tau_N}}{1-\beta} L_{max}$, where L_{max} denotes the maximum instantaneous cost and τ_N is a model dependent parameter that is $N \leq \tau_N$.*

4.2.3 RL algorithm for MDP Δ_N

Let $\mathcal{T}$ be a generic (model-based or model-free) RL algorithm designed for finite-state MDPs with infinite horizon discounted cost. By a generic RL algorithm, we mean any algorithm which fits to the following framework. At each iteration $k \in \mathbb{N}$, $\mathcal{T}$ knows the state of system, selects one action, and observes an instantaneous cost and the next state. The strategy learned by $\mathcal{T}$ converges to an optimal strategy as $k \to \infty$.

Let $\tilde{\boldsymbol{\psi}}_N^k : \mathcal{S}_N \to \mathcal{G}$ be the learned strategy associated with RL algorithm $\mathcal{T}$ operating on MDP Δ_N at iteration k such that

$$\lim_{k\to\infty} |\tilde{J}_N(\tilde{\boldsymbol{\psi}}_N^k) - \tilde{J}_N(\tilde{\boldsymbol{\psi}}_N^*)| = 0. \tag{8}$$

Now, we convert (translate) the strategies in Δ_N to strategies in the original multi-agent system described in Section 2, where the actual learning happens. Hence, we define a strategy $\boldsymbol{g}_N^k := (g_N^{k,i}, \ldots, g_N^{k,n})$, at iteration k, as follows:

$$g_N^{k,i}(s, m^i) := \tilde{\boldsymbol{\psi}}_N^{k,i}(s)(m^i), \forall s \in \mathcal{S}_N, \forall m^i \in \mathcal{M}^i, \forall i, \tag{9}$$

where $\tilde{\boldsymbol{\psi}}_N^{k,i}$ denotes the ith term of $\tilde{\boldsymbol{\psi}}_N^k$ and state s updates according to (7).

Theorem 2 *Let J^* be the optimal performance of the original multi-agent system given in* (4)*. Then, the approximation error associated with using the learned strategy is bounded as follows:*

$$\lim_{k\to\infty} |J^* - J(\boldsymbol{g}_N^k)| = |\tilde{J}(\tilde{\boldsymbol{\psi}}^*) - \tilde{J}_N(\tilde{\boldsymbol{\psi}}_N^*)| \leq \epsilon_N, \tag{10}$$

where $\epsilon_N = \frac{2\beta^{\tau_N}}{1-\beta} L_{max} \leq \frac{2\beta^N}{1-\beta} L_{max}$. Note that the error goes to zero exponentially in N.

5 Example

Consider a 2-user multi access broadcast channel (MABC) system first defined in [4]. The system consists of 2 users that have a buffer of size 1 (thus, $X_t = (X_t^1, X_t^2) \in \{0,1\}^2$). Packets arrive at each user i according to independent Bernoulli processes with rate p^i. Each user observes the state of its own queue i.e. $(Y_t^i = X_t^i)$ and transmits if it has a packet (i.e. $U_t^i \in \{0,1\}$ and $U_t^i \leq X_t^i$). If only one user transmits, then the transmission is successful and the packet is removed from the queue. If both users transmit, there is a "collision" and the packets remain in the queues. Users can sense whether the channel was used or if a collision took place. Thus, the information available at each user i is $I_t^i = \{X_t^i, \mathbf{U}_{1:t-1}\}$, where $\mathbf{U}_t = (U_t^1, U_t^2)$. The objective is to maximize the throughput. Hence, the instantaneous reward is defined as follows: $r(X_t, \mathbf{U}_t) = U_t^1 + U_t^2 - 2U_t^1 U_t^2$.

At time t, the common observation $Z_t = \mathbf{U}_t$ and the common information $C_t = \{\mathbf{U}_{1:t-1}\}$. For this specific model, the prescription γ^i is completely specified by $A_t^i := \gamma_t^i(1)$ (since $\gamma_t^i(0)$ is always 0). Hence, $U_t^i = \gamma_t^i(X_t^i) = A_t^i \cdot X_t^i$. Therefore, we may equivalently assume that the coordinator generates actions $\mathbf{A}_t = (A_t^1, A_t^2)$. Define $\mathbf{\Pi}_t = (\Pi_t^1, \Pi_t^2)$, $\Pi_t^i = \mathbb{P}(X_t^i = 1 \mid \mathbf{U}_{1:t-1}, \mathbf{A}_{1:t-1})$, as information state for the coordinated system with initial state $\mathbf{\Pi}_1 = (p^1, p^2)$. Thus, the reachable set $\mathcal{R}$ is given by $\mathcal{R} := \{(1,1), (1,p^1), (p^2,1), (p^1,p^2)\} \cup \{(p^1, T_2^n p^2) : n \in \mathbb{N}\} \cup \{(T_1^n p^1, p^2) : n \in \mathbb{N}\}$, where $T_i^n q = T_i(T_i^{n-1} q)$. Let b_1, b_2 be any arbitrary number in $(0,1)$. Define $\mathcal{S} = \{S_k\}_{k=1}^{\infty}$ as the countable state space of Δ, where $S_1 = \{(0,0)\}$ and $S_k = \{(0,0), (0,1), (1,0), (1,1), (0,1-b_1^i), (1-b_2^i, 0)\}_{i=1}^{k-1}$, $k \geq 2$. The action space is $\mathcal{A} = \{(0,1), (1,0), (1,1)\}$ (note that the action $(0,0)$ is dominated, so it is removed without loss of optimality).

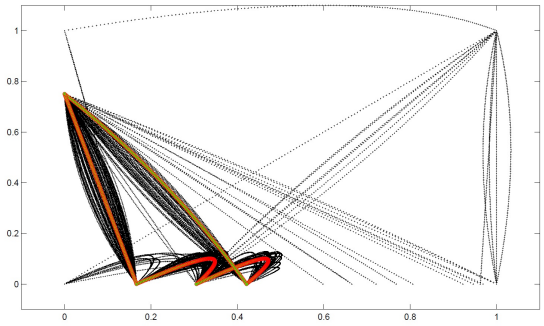


Figure 1: This figures shows the learning process of MDP Δ_N in a few snapshots. In this simulation, we use the following numerical values: $b_1 = 0.25, b_2 = 0.83, N = 20, \beta = 0.99, p^1 = 0.3, p^2 = 0.6$. In particular, the optimal strategy is a recurrent class consisting of states $(0, 1-b_1^1)$, $(1-b_2^1, 0)$, $(1-b_2^2, 0)$, and $(1-b_2^3, 0)$. The learning procedure is plotted in black and the optimal recurrent class is plotted in red. It is seen that the state of the system is eventually trapped in the optimal recurrent class. The optimal strategy says that user with rate of 0.6 must transmit 3 times more than the user with rate of 0.3.

References

[1] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis, Decentralized Stochastic Control with Partial History Sharing: A Common Information Approach, IEEE Transaction on Automatic Control, vol. 58, no. 7, 2013.

[2] Tilak, Omkar and Mukhopadhyay, Snehasis, Partially decentralized reinforcement learning in finite, multi-agent Markov decision processes, AI Communications, vol. 24, no. 4, pp 293–309, 2011.

[3] Busoniu, Lucian and Babuska, Robert and De Schutter, Bart, A comprehensive survey of multiagent reinforcement learning, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol.38, no. 2, pp 156–172, 2008.

[4] M. G. Hluchyj and R. G. Gallager, Multiacces of a slotted channel by nitely many users, Proc. of Nat. Tel. Con., pp. 421-427, 1981.

[5] M. T. J. Spaan, N. Vlassis, and F. C. A. Groen, High level coordination of agents based on multiagent Markov decision processes with roles, in Workshop on Coop. Rob., IEEE/RSJ Int., pp. 66-73, 2002.

[6] N. Vlassis, A concise introduction to multiagent systems and distributed AI,Univ. of Amsterdam, Tech. Rep., 2003.

[7] J. R. Kok, M. T. J. Spaan, and N. Vlassis, Non-communicative multirobot coordination in dynamic environment, Robotics and Autonomous Systems, Vol. 50, No. 2-3, pp. 99-114, 2005.

[8] C. Claus and C. Boutilier, The dynamics of reinforcement learning in cooperative multiagent systems, 10th Conf. on Inn. Appl. of AI , pp. 746-752, 1998.

[9] S. Kapetanakis and D. Kudenko, Reinforcement learning of coordination in cooperative multi-agent systems, 14th conf. on Inn. Appl. of AI , pp. 326-331, 2002.

[10] Laetitia Matignon, Guillaume J. Laurent and Nadine Le Fort-Piat, Hysteretic Q-Learning : an algorithm for Decentralized Reinforcement Learning in Cooperative Multiagent Teams, Int. Rob. and Sys., 2007.

[11] Jing huang, Bo Yang, and Da-You Liu, A distributed Q-learning Algorithm for Multi-Agent Team coordination, IEEE 40th Int. conf. on Mach. Ler. and Cyb., Vol. 1, pp. 108-113, Aug., 2005.

Model-based Analysis of the Tower of London Task

Constantinos Mitsopoulos
Centre for Brain and Cognitive Development
Centre for Cognition, Computation and Modelling
Department of Psychological Sciences
Birkbeck, University of London, WC1E 7HX
c.mitsopoulos@bbk.ac.uk

Denis Mareschal
Centre for Brain and Cognitive Development
Centre for Cognition, Computation and Modelling
Department of Psychological Sciences
Birkbeck, University of London, WC1E 7HX
d.mareschal@bbk.ac.uk

Richard Cooper
Centre for Cognition, Computation and Modelling
Department of Psychological Sciences
Birkbeck, University of London, WC1E 7HX
r.cooper@bbk.ac.uk

Abstract

The planning process is central to goal-directed behaviour in any task that requires the organization of a series of actions aimed at achieving a goal. Although the planning process has been investigated thoroughly, relatively little is known about how this process emerges and evolves during childhood. In this paper we describe three reinforcement learning models of planning, in the Tower of London (ToL) task, and use Bayesian analysis to fit each model to pre-existing data from 3-4 year-old and 5-6 year-old children performing the task. The models all capture the increased organisation seen in the older children's performance. It is also shown that, at least for this dataset, the most complex model – that with discounting of future rewards and pruning of highly aversive states – provides no additional explanatory power beyond a simpler discounting-only model. Insights into developmental aspects of the planning process are discussed.

Keywords: Reinforcement Learning, Shaping Rewards, Planning, Tower of London

Acknowledgements

We are deeply indebted to Peter Dayan for all the discussions that inspired this work. This research was supported by the European Commission grant MC-ITN-289404-ACT.

1 Introduction

Environmental stimuli combined with external rewards or punishments may elicit certain responses, which ultimately lead to learned behaviours. In this context, extrinsic motivation, which means to be moved to do something because of a specific reward outcome, may be distinguished from intrinsic motivation, which means to be moved to do something because it is inherently enjoyable [1]. Intrinsic motivation is evident in animal behaviour, where it has been found that organisms engage in exploratory, playful and curiosity-driven behaviour even in the absence of an environmental reinforcement [4]. Similarly, researchers in many areas in cognitive science emphasize the importance of intrinsically motivated behaviour in human development and learning.

Our concern in this paper is whether instrinsic motivation might play a role in the cognitive processes underlying planning. Human planning has been studied extensively using look-ahead puzzles, in which subjects have to preplan mentally a sequence of moves in order to transform a starting configuration of the puzzle to a goal configuration, subject to a set of rules. In the Tower of London (TOL; [7]) task, for example, subjects are required to rearrange three balls on three pegs so that the configuration of balls matches a goal state (see figure 1), but in doing so they must adhere to a set of rules or constraints. Thus they must move only one ball at a time, and place it back on a peg before moving another ball. The task can be viewed as a sequential decision making puzzle, with reward obtained if / when the player achieves the goal state. Here, however, we use computational methods to explore the effects of more frequent feedback – reflecting intrinsic motivation – on appropriate moves that may guide the subject towards solving the puzzle. Specifically, we model an existing dataset from children's planning on the ToL by incorporating a *reward shaping function*, representing the intrinsic motivation of the child, within the framework of model-based reinforcement learning.

Figure 1: A typical Tower of London problem. The task consists of a board with three pegs, each one with different heights, and three different coloured balls. The right peg can contain up to three balls, the middle peg up to two balls and the left one only one ball. The balls are initialy arranged in one configuration on the pegs and the goal is to move balls – one at a time and from peg to peg – in order to achieve the given goal configuration. The problem shown requires 3 moves, but more difficult problems may require up to 7 moves.

2 Modelling the ToL task

In problems such as the ToL, the goal is achieved by decomposing it into subgoals and evaluating the order of simple moves towards the goal [3]. It is this evaluation procedure that guides our approach to planning in such a task. We model children's behaviour on the ToL as a Markov Decision Process (MDP) and follow model-based approaches.

2.1 The Extended State Space of ToL

Within the empirical study on which this work is based (described in more detail below), some children (especially the younger ones) failed to adhere to the task rules. That is, although it was explained to each child that he/she should only move one ball at a time using only one hand, and although the child in each case claimed to understand this restriction and demonstrated this knowledge in a series of practice trials, sometimes he/she would pick up one ball in one hand in order to reorder the position of the other two balls that would otherwise require a series of moves. This typically occurred when the state of the apparatus almost matched the goal state, with two balls on one peg being in the wrong order (e.g., red immediately above blue when blue should have been immediately above red). One possibility in this case is that the child's look-ahead process suggests to him/her that there is great similarity between the current and goal states, yet any (legal) move would result in a decrease in similarity. From the perspective of search through a decision tree, pruning of the tree might take place when facing such situations, leaving the child with only one viable option – to move both balls at the same time and hence break the task rules.

In order to accommodate breaking the task rules by subjects, we expand both the state space (from 36 states to 114 states) by adding two more locations representing the hands of the child (effectively two additional pegs, each of which can hold at most one ball), and the set of available actions (adding actions corresponding to moving balls to and from the hands). This yields an extended state transition matrix $T : S \times A$ with $[114 \times 25]$ entries.

2.2 The Reward Function

The design of the transition matrix is straightforward, as the task is deterministic, but for the reward function further assumptions are necessary. In the Tower of London task, the reward from the environment is given to the subject only at the goal state. In addition to this, however, we assume that subjects are driven step-by-step towards the goal state by an internal reward function, which is

related to the similarity of the current configuration of the task, state s_t at time t, to the desired configuration (i.e., the goal state). By "similarity" we mean the degree of overlap, in terms of positions of the balls at the pegs, between two states (as defined in the following paragraph). In other words, in the planning process we assume that subjects evaluate their future actions in terms of not just whether they achieve the goal state, but (for other states) how close they bring them to the goal state. Previous work has shown that such a modification to the reward structure often suffices to render straightforward otherwise intractable learning problems. Additionally, proper modification can leave the optimal strategy invariant (see [6]).

To calculate state similarity, as required by the internal reward function, we represent each state within the ToL by a set of 24 binary features (bits). For each ball we assign three bits to represent its vertical position on the peg and five bits for the peg that the ball is placed on (three for the real pegs and two representing the hands). According to this scheme if the red ball is at the lowest position on the first peg then it will be represented as $R_{pos} = (1,0,0)$ and $R_{peg} = (1,0,0,0,0)$. The state vector is the concatenation of the vectors for each ball: $\mathbf{s}_t = (R_{pos}, R_{peg}, G_{pos}, G_{peg}, B_{pos}, B_{peg})$. We then define the *similarity between two states* as the inner product between those states. The reward shaping function therefore has the form $F(s,s') = \phi(\mathbf{s}') - \phi(\mathbf{s})$ where $\phi(\mathbf{x}) = \mathbf{x}^T \mathbf{x}_{goal}$ is the inner product between the state representations under comparison (with bold letters denoting the state vector of features).

It seems that some children perceive some configurations ("towers" where all three balls are on the longest peg, or "flats" where all three balls are on different pegs) as being the same, independent of the arrangement of colour. The above approach helps us capture similarity in the structure of a particular configuration (i.e., the number of balls on each of the pegs is the same for both configurations independent of colour).

3 Methods

3.1 Model-based analysis

In this section we describe three model-based RL models used to describe the planning process. The models, the model fitting and model comparison procedures are described in detail in [5] but we repeat the description here for completeness. All three models assume that subjects choose actions stochastically, with the probability of choosing action (or choice) c from state s at time t given by:

$$p(c_t|s_t) = \frac{e^{\beta Q(s_t,c_t)}}{\sum_{c'} e^{\beta Q(s_t,c')}} \tag{1}$$

The parameter β is an inverse temperature that represents the subject's sensitivity to rewards. The three models differ in the calculation of the function $Q(s_t,c_t)$.

The first model is the *Lookahead* model. This model is simply a tree search model in the sense that searches all available options until the end of the tree:

$$Q_{lo}(s,c) = R(s,c) + \max_{c'} Q_{lo}(s',c') \tag{2}$$

where s' is the successor state from state s by selecting choice c. In all the models we set $R(s,c) = (1-w)R_{ext} + wR_{int}$. This means that a low w will indicate goal directed behaviour whereas high w indicates planning driven by state similarity.

For the extended ToL with 25 available actions at each state[1], and a decision tree of depth $D = 3$, the total number of action choices considered by the lookahead model is 16275. This number is large and children are unlikely to evaluate this number of actions during planning. One possibility is that they prune the decision tree and evaluate action trajectories according to their expected outcome. This leads to the second model, the *Discount* model.

We assume that at each depth of the decision tree, a biased coin is flipped in order to determine whether the tree search should be terminated and return zero reward. Let the probability of stopping be γ , and using a mean-field approximation (in order not to use the immense number of possible choices at the branches of the decision tree), the Q values are estimated by:

$$Q_d(s,c) = R(s,c) + (1-\gamma)\max_{c'} Q_d(s',c') \tag{3}$$

Then the future outcome, k steps ahead, is weighted by the probability $(1-\gamma)^{k-1}$ that it is encountered.

The third model we used is a modification of the Discount model, and we refer to this as the *Pruning* model. We test the hypothesis that subjects avoid states with great dissimilarity with the goal state. Thus we modify the calculation of Q from the Discount model to the following:

$$Q_{pr}(s,c) = R(s,c) + (1-x)\max_{c'} Q_{pr}(s',c') \tag{4}$$

where

$$x = \begin{cases} \gamma_S & \text{if } R_{int}(s,c) \text{ is a large dissimilarity} \\ \gamma_G & \text{else} \end{cases} \tag{5}$$

[1]The actual available choices, at each state are given by counting all possible transitions of the balls at the pegs, including the two extra pegs which represent the hands of the child.

γ_S (Specific pruning parameter) is the probability that the subject stops evaluating the decision tree at a state where the immediate reward leads to a state with great dissimilarity with the goal state. γ_G (General pruning parameter) is γ as in the Discount model.

3.2 Model fitting procedure

We assume an hierarchical model on how data are generated for each age group. Each model is characterized by a set of parameters $\mathbf{k}_i$, for each subject i, that are generated by a Gaussian distribution $\mathbf{k}_i \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{v}^2)$ with parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \mathbf{v}^2)$. We will refer to these as *hyperparameters*. The whole analysis is applied separately for each of the two age groups. We fit the model parameters and the hyperparameters in a joint scheme, using the EM algorithm [2], maximising the marginal likelihood given all data by all N subjects:

$$\hat{\boldsymbol{\theta}}^{ML} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{C}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \left(\prod_i^N \int P(\mathbf{c}_i|\mathbf{k}_i) P(\mathbf{k}_i|\boldsymbol{\theta}) d^N \mathbf{k}_i \right) \tag{6}$$

where $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^N$ is the set of all actions performed by each subject i. Actions are assumed to be independent. Thus they factorize over trials. According to the above, for the E-step at the j^{th} iteration we use the Laplace approximation to approximate the integral of the marginal (eq. 6) and the parameters are estimated at the maximum of the posterior distribution (MAP). For the M-step we estimate the hyperparameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \mathbf{v}^2)$, by maximising the expectation computed at the E-step. For the Lookahead model we fitted 1 parameter, for the Discount model 2 parameters, and for the Pruning model 3 parameters. All parameters were transformed before inference to enforce constraints ($\beta \geq 0, 0 \leq \gamma_S, \gamma_G \leq 1$). The above procedures were verified by using surrogated data from a known decision process.

3.3 Model comparison

Given the three models, and given that the models have different number of parameters, it is important to compare the models to understand which best accounts for the children's behaviour. Having no prior knowledge about each model, we assume that models are equally likely a priori. Thus we examine only the log likelihood of each model $\log P(\mathcal{C}|\mathcal{M})$. This quantity can be approximated by the Bayesian Information Criterion (BIC) as:

$$\log P(\mathcal{C}|\mathcal{M}) = \int P(\mathcal{C}|\boldsymbol{\theta}) P(\boldsymbol{\theta}|\mathcal{M}) d\boldsymbol{\theta} \approx -\frac{1}{2}\mathrm{BIC}_{int} = \log P(\mathcal{C}|\hat{\boldsymbol{\theta}}^{ML}) - \frac{1}{2}|M| \log |\mathcal{C}| \tag{7}$$

where $|\mathcal{C}|$ is the total number of choices made by all subjects of the group being examined, and $|\mathcal{M}|$ is the number of prior parameters (mean and variance for each parameter) that we estimated empirically above. The first term on the right hand side of equation 7 was estimated by standard Monte Carlo approximation. The Bayesian Information Criterion (BIC_{int}), apart from penalizing the model for extra parameters, is not the sum of individual likelihoods, but the sum of *integrals* over individual parameters thus the *int* (integral) subscript. With this approach we compare not only how well a model fits the data when its parameters are optimized, but also how well a model fits the data when we use information about where the group level parameters lie on average [5].

Although the above gives a good comparative measure of model fit, an absolute measure is needed in order to ensure that the best model does indeed describe the data generation procedure efficiently. Given the MAP estimation of each subject's parameters, we compute the mean total "predictive probability" for subjects N, in a number of trials T, which is the geometric mean of all the $P(c_t|s_t, \mathbf{k}_i)$ where c_t is the action selected at trial t by the i^{th} subject, at the state s_t from a decision process parametrized by $\mathbf{k}_i$.

4 Experimental results and discussion

To evaluate the three planning models we consider existing data from seventeen 3-4 year olds (mean age 47 months) and seventeen 5-6 (mean age 68 months) years old on six ToL problems of graded difficulty [8]. The younger children in this study struggled to complete many of the problem, and in both groups some children failed to complete all problems. Therefore in the analysis below we restrict attention to data from 10 of the younger children and 13 of the older children. Given the population and the number of problems we obtained 60 and 78 action sequences for younger children and older children respectively. Among younger children, 7 out of 10 performed illegal moves (37 total), whereas 5 out of 13 of the older children used illegal moves (22 total). As illegal moves we count the transition of a ball from a peg to the hand and not the other way around. The original dataset and experimental conditions are described in detail in [8] and summarised in Table 1.

The phenomenon of the direction of the behaviour towards a perceptual match with the goal state (i.e., reaching a state with the same configuration of the balls as the goal state, except the colour of the balls is different in the goal state, and declaring that the goal state reached) is much evident in younger children. The inferred parameter w was 0.28 and 0.52 (Pruning Model estimation) for older and younger children respectively, revealing a significant difference between the planning mechanisms between the two groups. This suggests that younger children pursue a similarity match between goal state and their current state whereas the older children follow more goal directed behaviour. However, by comparing BIC_{int} scores (e.g., 5-6yrs old group: Lookahead (1455), Discount (1105), Pruning (1115)) and mean predictive probabilities (e.g., 5-6yrs old group: Lookahead (0.85), Discount (0.89), Pruning (0.89)), we found that the Discount and Pruning models describe children's behaviour better than the Lookahead model,

Table 1: The percentage of children showing different behaviours in each age group

Sequences	3-4yrs	5-6yrs
Reached goal correctly	38.3%	64.1%
Reached goal with illegal moves	40.0%	24.3%
Perception matching	20.0%	5.1%
Interrupted/Stopped	1.7%	6.4%

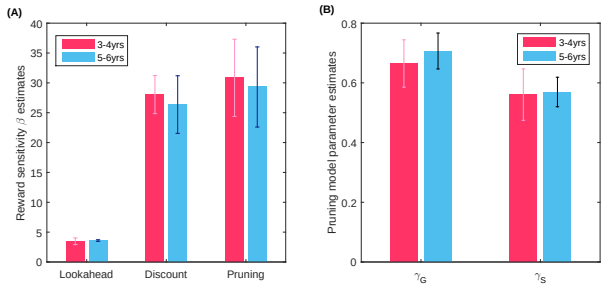


Figure 2: Model parameters estimates: (A) Reward sensitivity β estimates from the three models. (B) Estimations of General pruning γ_G and Specific pruning γ_S parameters for the two age groups given the pruning model.

though the extra parameter of the Pruning model does not improve the model predictions beyond that of the Discount model (at least in the specific ToL problems tested). This may reflect a lack of sophistication in planning ability at these ages. Further investigation of behaviour on specific ToL problems could reveal the importance of various features that affects their planning process.

An analysis of choice behaviour according to our models shows that older children prune, in general, more than the younger children. However the difference is very small. This early termination of the decision tree for the younger ones, appears to be mainly because are driven by the (perceived) similarity of the current state to the goal state, leading to them "cheating" by holding two balls at the same time In addition, younger children tend to overprune their decision tree and confuse the objective similarity between states. On the other hand older children demonstrated a better level of planning (i.e., reaching the goal state by following the rules consistently). They tend to prune but in a way that leads them to the goal state without shortcuts (i.e., without picking up more than one ball at a time). Furthermore, the older children tended not to show confusion in distinguishing very similar states. Finally looking at the reward sensitivity parameter β (Fig 2A), younger children are much more greedy to rewards than older children pursuing a perceptual match between current state and goal state.

We have demonstrated a method for analysing human behaviour in puzzle tasks where the main reward factor is the internal reward, represented by a shaping reward function. By testing it in a real world example as the above, useful insights can be gained concerning differences in mental planning between age groups, though further work needs to be conducted to formally explore the relationship between internal reward representations and planning in child development.

References

[1] E L Deci and R M Ryan. *Intrinsic Motivation and Self-Determination in Human Behavior*. Springer Science & Business Media, 1985.

[2] A P Dempster, N M Laird, and D B Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[3] K J Gilhooly, L H Phillips, V Wynn, R H Logie, and S Della Sala. Planning processes and age in the five-disc tower of london task. *Thinking & reasoning*, 5(4):339–361, 1999.

[4] H F Harlow. Learning and satiation of response in intrinsically motivated complex puzzle performance by monkeys. *Journal of comparative and physiological psychology*, 43(4):289–294, 1950.

[5] Q J M Huys, N Eshel, E O'Nions, L Sheridan, P Dayan, and J P Roiser. Bonsai trees in your head: how the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS computational biology*, 8(3):e1002410, 2012.

[6] A Y Ng, D Harada, and S J Russell. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. *International Conference on Machine Learning*, 16:278–287, 1999.

[7] T Shallice. Specific Impairments of Planning. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 298(1089):199–209, 1982.

[8] R Waldau. A developmental study of problem solving in children aged 3–6: Development of planning strategies, 1999. Undergraduate disertation.

Approximate Linear Successor Representation

Clement A. Gehring
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institutes of Technology
Cambridge, MA 02139
gehring@csail.mit.edu

Abstract

The dependency of the value function on the dynamics and a fixed rewards function makes the reuse of information difficult when domains share dynamics but differ in their reward functions. This issue is present when attempting to reuse options to solve different tasks in the same domain. If instead of a value function, a successor representation is learned for some fixed dynamics, then any value function defined on any linear reward function can be computed efficiently. This setting can be particularly useful for reusing options in some hierarchical planning settings. Unfortunately, even linear parametrization of successor representation require a quadratic number of parameters with respect to the number of features and as many operations per temporal difference update step. We present a simple temporal difference-like algorithm for learning an approximate version of the successor representation with an amortized runtime $O(nk)$ for n features and the maximum rank-k of the approximation. Preliminary results indicate that the rank parameter can be much smaller than the number of features.

Keywords: successor representation, occupancy function, universal option model, incremental singular value decomposition, temporal difference

Acknowledgements

This work was supported in part by the NSF (grant 1420927). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also gratefully acknowledge support from the ONR (grant N00014-14-1-0486), from the AFOSR (grant FA23861014135), and from the ARO (grant W911NF1410433).

1 Introduction

Hierarchical planning is a necessity for solving very large problems by breaking the problem into many short-horizon sub-problems. Options, a reinforcement learning (RL) formulation of macro-actions, can allow an agent to plan at a more abstract level [7]. In order to efficiently plan with options, it is necessary to build option models predicting the total reward from the start of an option in a given state to its completion. However, simply learning a value function for every option might not allow for the amount of generalization an efficient agent requires. This is in part due to the dependency of value functions on both the dynamics of the problem and the reward function.

As motivations for this work, we can think about a specific kind of hybrid RL planner. One can imagine a series of sub-tasks (i.e., reward functions), generated by an high-level planner, which are fed to an RL agent. In this context, we wish the RL agent to generate, off-line, and efficiently, new policies for the sub-tasks. This can be done with pre-defined options and their learned models. However, since every sub-task uses a different reward function, we need a generalized version of these models as proposed by Yao [8] in order to build models invariant to the reward function. Given these tools, the RL agent could then use model-based methods (e.g., dyna [4]) to efficiently find a good policy. This generalization of option models is achieved by learning a successor representation in lieu of value functions [3]. Successor representations (SR), which predict the expected discounted times future states are visited, have recently gained traction as models for animal behaviour. They offer an interesting addition to the model-free learning process (the *habitual* system). In addition, there seems to exist interesting shared characteristic between the successor representation and place cells as wells as well as the egiendecomposition of the SR and grid cells [5]. These results suggest that a compressed version of the SR might be useful for planning.

The SR solve an important issue with re-using options but come with a quadratic memory and computational cost. These drawbacks limit their usefulness as a tool for efficient planning. In this work, we present an approximate linear successor representation algorithm which would allow universal option models to be learned in much bigger feature spaces. In order to learn a rank-k approximation on n features, our temporal difference-like algorithm has an amortized cost $O(k^2 + nk)$ and requires $4nk + k$ parameters. This allows new value functions to be evaluated with a smaller matrix-vector product in $O(nk)$. Preliminary results indicate that k can be set much smaller than n. Furthermore, the performance degrades gracefully as the rank decreases, allowing a designer to choose a trade-off between computation time and accuracy.

We begin in Section 2 by presenting the reinforcement learning framework used as well as the definition of the successor representation. In Section 3, we define the rank-k approximation of the successor representation, after which, we present a novel algorithm for incrementally learning an approximate successor representation with a series of singular value decompositions. We present some preliminary results in Section 4 demonstrating the correctness of this approach in a small domain. We conclude with a discussion of the promising future work this approach might lead to. For brevity, we present the approximate successor representation algorithm as a standalone algorithm but the reader should keep in mind that this approach can easily be adapted to replace the "*accumulation* part" from Yao's work on learning universal option models [8].

2 Background

We model the problem as a Markov Decision Process (MDP) with states $\mathbb{S}$, actions $\mathbb{A}$, transition probabilities $\mathcal{P}^a(s, s')$, and reward function $r(s, a)$ for s and $s' \in \mathbb{S}$ and $a \in \mathbb{A}$. We consider actions sampled from a fixed policy π inducing transition probabilities $\mathcal{P}^\pi$. We seek to represent value functions V^π defined as

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \;\middle|\; S_0 = s\right],$$

where R_t is the reward received at time t, and $\gamma \in [0, 1)$ is a discount factor. Given that we only consider the fixed policy case, we omit the π superscript but it is important to keep in mind that the value function always depends on the policy.

In some cases, the states are discrete and few in number, allowing us to keep a separate value for each. We refer to this as the tabular case. However, in most interesting applications, the state space is large and continuous. In order to handle this case, we consider approximating the value functions with a linear combination of basis functions. This can be written as

$$\hat{V}(s) = \theta^T \phi(s),$$

where $\phi : \mathbb{S} \to \mathbb{R}^n$ corresponds to some feature representation of the state, and $\theta \in \mathbb{R}^n$ is a parameter vector. The successor representation is defined as the discounted total number of times a state is seen after starting from any other state [3]. More formally, in the tabular case, we define a matrix F, so that element $[F]_{i,j}$ denotes the expected discounted sum of indicator random variables with value 1 if the agent is in state j at that time, given that the trajectory starts in

state i.

$$[F]_{ij} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{S_t=j} \;\middle|\; S_0 = i\right],$$

where $\mathbb{1}_{S_t=j}$ has value one when the state at time t is j and zero otherwise. The successor representation can be naturally extended to linear function approximation in the following way

$$\phi(s)^\top \theta_i \approx \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t [\phi(S_t)]_i \;\middle|\; S_0 = s\right]$$
$$\Theta \triangleq [\theta_1 \quad \theta_2 \quad \dots \quad \theta_n],$$

where θ_i is the vector of parameters for a linear value function where feature i is accumulated instead of the rewards, and Θ is a concatenation of the columns θ_i. This means that $\phi(s)^\top \Theta$ is a row vector in which entry j is, approximately, the expected discounted accumulation of feature j's future value. It is important to note that the successor representation is invariant to the reward function. Computing the successor representation can be particularly helpful, as it allows us to easily compute any linear value function with arbitrary linear reward function (sharing the same transition probabilities). For a reward function of the form $r(s,a) = w^\top \phi(s)$, we compute a corresponding value function as $V(s) = \phi(s)^\top \Theta w$.

Once Θ is obtained, all that is required is to estimate w through linear regression, which is significantly easier to do than fitting a value function as it doesn't require any temporal abstraction. In addition, in many cases, it makes sense to assume w is given.

Since each column of Θ is analogous to a value function, the parameters Θ_t at time t can be updated using temporal difference with eligibility traces [6], giving the following update rule for a sample transition $\{s_t, r_t, s_{t+1}\}$:

$$\begin{aligned} e_t &\leftarrow \gamma\lambda e_{t-1} + \phi(s_t) \\ \delta_t &\leftarrow \phi(s_t) + \gamma\Theta_t^\top \phi(s_{t+1}) - \Theta_t^\top \phi(s_t) \\ \Theta_{t+1} &\leftarrow \Theta_t + \alpha e_t \delta_t^\top, \end{aligned} \tag{1}$$

where δ_t is a vector of temporal difference errors (for each feature), and e_t is an eligibility trace vector allowing updates to propagate through recently visited states. This results in update steps requiring $O(n^2)$ operations per complete update.

3 Approximate Linear Successor Representation

Since the successor representation is inherently quadratic in size, a natural approach is to consider some form of compression. Unfortunately, it is known that in the tabular case $F = (\mathbb{I} - \gamma\mathcal{P}^\pi)^{-1}$ and is therefore full rank. Instead, we consider a rank-k approximation of Θ of the form $\hat{\Theta} = U\Sigma V^\top$, where Σ is a $k \times k$ diagonal matrix and U and V are both $n \times k$ orthogonal matrices. The optimal such approximation (with respect to the reconstruction error) would be the singular value decomposition (SVD) of Θ. For this reason, we will consider an incremental SVD algorithm with truncation. The truncation will cause the algorithm to be sub-optimal (i.e., the result might not be the true SVD) but, empirically, this hasn't caused any problems for large enough k. In our experiments, we have observed the singular values of Θ decay very quickly which might suggest that k can often be set much smaller than n.

Instead of keeping a set of parameters Θ, our algorithm maintains five matrices, U, Σ, V, A, and B. The matrices U, Σ, V carry the same meaning as before. We include two extra $n \times k$ matrices which serve as temporary storage for updates. These buffers allow us to defer updates of the SVD in order to improve the runtime as well as the numerical accuracy. At all times, we can compute the most recent estimate of Θ as $\hat{\Theta} = U\Sigma V^\top + AB^\top$. The matrix-vector product $\hat{\Theta}^\top \phi(s)$ can be evaluated as a series of $n \times k$ matrix-vector operations. This algorithm is never required to build the quadratic $\hat{\Theta}$. Since every temporal-difference learning update has the form of an outer product, e_t's are stored as columns of A and δ_t's as columns of B. This allows updates of the SVD to be deferred while still updating the values of $\hat{\Theta}$. We store k updates in A and B before updating the SVD and zeroing A and B.

We use a simple algorithm presented in Brand et al. [2] for updating the SVD. It requires two QR decompositions on $n \times k$ matrices and one SVD on a $2k \times 2k$ matrix, as shown in Algorithm 1. This results in an amortized cost of $O(k^2 + nk)$ per update. The full pseudocode for our approximate linear successor representation (ASLR) update step is given by Algorithm 2. Note that in the case where $k = n$, this algorithm is exact (given infinite numerical precision) and will give the true SVD of Θ since every update step is identical to (1).

4 Experimental results

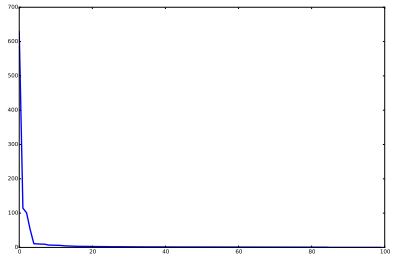
We offer a set of preliminary results on the behaviour of rank-k approximations of the successor representation. We refer to the temporal-difference algorithm for learning successor representation as SR and our approach as ALSR. In

Algorithm 1 SVD update

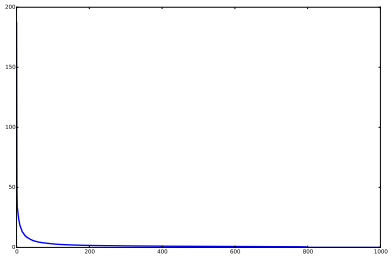
function UPDATE-SVD(U, Σ, V, A, B, k)
$Q_a, R_a \leftarrow$QR($(\mathbb{I} - UU^\top)A$)
$Q_b, R_b \leftarrow$QR($(\mathbb{I} - VV^\top)B$)
$K \leftarrow \begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix} + \begin{bmatrix}U^\top A\\ R_a\end{bmatrix}\begin{bmatrix}V^\top B\\ R_b\end{bmatrix}^\top$
$U', \Sigma', V' \leftarrow$ SVD(K)
$U \leftarrow \begin{bmatrix}U & Q_a\end{bmatrix} U'$
$V \leftarrow \begin{bmatrix}V & Q_b\end{bmatrix} V'$
$\Sigma \leftarrow \Sigma'$
return TRUNCATE(U, Σ, V, k)
end function

Algorithm 2 ALSR update

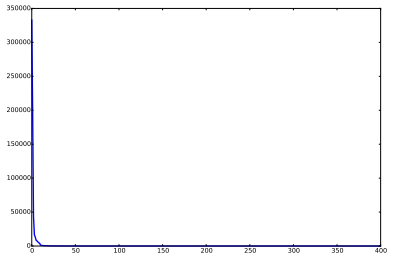
Given a sample transition $\{s_t, r_t, s_{t+1}\}$
$e_t \leftarrow \gamma\lambda e_{t-1} + \phi(s_t)$
$\delta_t \leftarrow \phi(s_t) + \gamma(V\Sigma U^\top + BA^\top)\phi(s_{t+1})$
$\quad -(V\Sigma U^\top + BA^\top)\phi(s_t)$
if $i \geq k$ **then**
$U, \Sigma, V \leftarrow$ UPDATE-SVD(U, Σ, V, A, B, k)
$A \leftarrow 0^{m\times k}$
$B \leftarrow 0^{m\times k}$
$i \leftarrow 0$
else
$A_{1:m,i} \leftarrow e_t$
$B_{1:m,i} \leftarrow \delta_t$
$i \leftarrow i + 1$
end if



(a) The sorted singular values of Θ in the 4-room domain with tabular representation and random actions. The successor representation was estimated from 1000 episodes each of 1000 steps.



(b) The sorted singular values of Θ in the Moutain car domain with a 20×20 radial basis encoding plus a bias term. The successor representation was estimated from 4000 episodes of experience following an energy pumping policy.



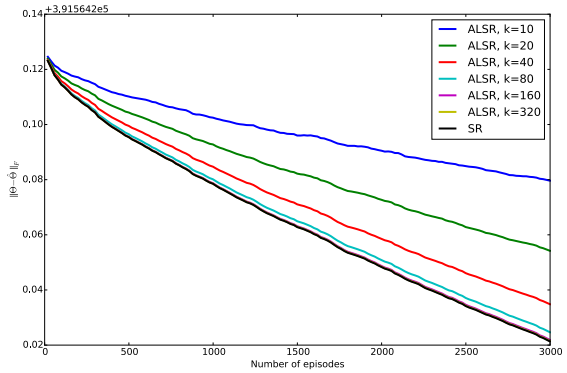
(c) The sorted singular values of Θ in the Moutain car domain with 10 tilings discretizing the state in a 10×10 grid plus a bias term. The successor representation was estimated from 4000 episodes of experience following an energy pumping policy.

Figure 1: The sorted singular values of Θ

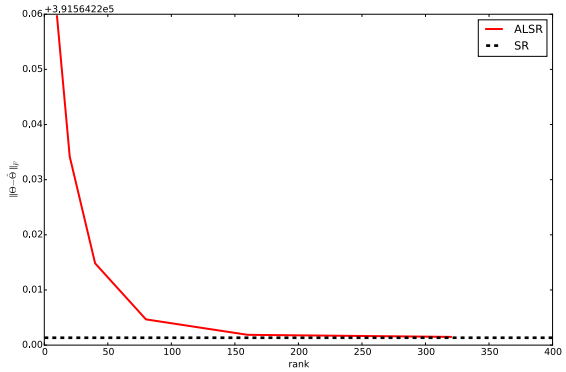
small domains, it is computationally feasible to approximate Θ fully using batched Least-Squares Temporal Difference (LSTD) [1] followed by a full SVD. In order to establish whether or not it is reasonable to approximate Θ with our approach, we plotted in Figure 1 the singular values of the successor matrix for the 4-room domain, defined in Sutton et al. [7], with tabular representation and random actions, and the Mountain car domain with a radial basis encoding and CMAC encoding under an energy pumping policy. These plots show that there are few important singular values. This result is encouraging since such a structure indicates that the matrix can be approximated with a relatively small rank.

We explored ALSR's performance more carefully on the Mountain car domain, in which an agent is tasked with driving a 1D under-actuated car back and forth in order to escape a valley. We used the dynamics described by Sutton and Barto [6], with random restart. The agent followed an energy pumping policy which always attempts to drive in the same direction as the velocity. With a 20×20 radial basis function encoding, we considered the effect of the parameter k with the approximated solution. With LSTD, we computed an empirical truth to which we compared $\hat{\Theta}$ by computing the Frobenius norm of their difference. Figure 2a shows the effect of the rank parameter, k, on the speed of convergence. Figure 2b highlights the effect of k on the solution after 3000 episodes. The dotted line represents the SR performance.

Our results in this experiment indicate that the performance of ALSR decreases as the rank is reduced, as one would expect. It is also encouraging to note that the same performance as SR is achieved with a rank significantly smaller than the number of features. This paves the way for much larger experiments where the computational gain of using ALSR could much greater. As a final observation, note that the required rank for achieving a reasonable performance with ALSR is higher than the required rank for a good approximation of Θ. This is most likely due to the greedy nature of the truncations in the incremental SVD. With our implementation, ALSR with $k = 40$ was an order of magnitude faster than SR when running on 900 features. This advantage decreased as the number of features was reduced, as expected. Though we haven't rigorously evaluated ALSR's runtime, this result hints that it would be advantageous in cases with a large number of features.



(a) The error of the estimated $\hat{\Theta}$ with respect to the number of episodes. The bottom lines represent the full successor representation and the approximate successor representation with high rank. Scores are averaged over 5 independent trials.



(b) The error of the estimated $\hat{\Theta}$ as a function of the rank parameter. The score represent the $\hat{\Theta}$ learned after 3000 episodes. Scores are averaged over 5 independent trials.

Figure 2: Estimation accuracy of Θ in the Mountain Car domain

5 Discussion

In this work, we proposed a novel approach for learning successor representation. Our ALSR algorithm is more efficient than the full SR temporal-difference algorithm and can potentially be applied to much large domains. We presented some preliminary results on small domains in which we have observed a graceful decay in performance as the rank is lowered and a comparable performance to that of the full successor representation.

Our results are preliminary and of limited scope. In the future, we hope to expand on this work by exploring ALSR's behaviour in larger domains, and settings with changing policies. In order to achieve better performance, we plan on adapting some of the modern temporal-difference algorithm variants to work with ALSR. This work focused on the successor representation used by the universal option model. There still remains a quadratic part in these models to optimize before they can be fully applied to large domains. It is possible that the model's expected next state functions can be similarly approximated by an incremental SVD approach. ALSR could easily be adapted to serve this goal in the future.

Finally, we would like to explore the link between the singular vectors found by ALSR and proto-value functions. The left singular vectors U can be seen as forming a basis over value function. It is possible that they can be used as proto-value functions (or could be equivalent in some way). This is an interesting avenue for upcoming research we hope to pursue.

References

[1] Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[2] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006.

[3] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[4] Jonathan Sorg and Satinder Singh. Linear options. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 31–38. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[5] Kimberly L Stachenfeld, Matthew Botvinick, and Samuel J Gershman. Design principles of the hippocampal cognitive map. In *Advances in Neural Information Processing Systems*, pages 2528–2536, 2014.

[6] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[7] Richard S Sutton, Doina Precup, and Satinder P Singh. Intra-option learning about temporally abstract actions. In *ICML*, volume 98, pages 556–564, 1998.

[8] Hengshuai Yao, Csaba Szepesvari, Richard S Sutton, Joseph Modayil, and Shalabh Bhatnagar. Universal option models. In *Advances in Neural Information Processing Systems*, pages 990–998, 2014.

Modeling Individual Differences in Risky Decision-Making with Cumulative Prospect Theory

Claire McCormick
Department of Psychology
University of Victoria
mccormic@uvic.ca

Meghann Pasternak
Department of Psychology
University of Victoria
meghannp@uvic.ca

Adam Krawitz
Department of Psychology
University of Victoria
akrawitz@uvic.ca

Abstract

Cumulative prospect theory (CPT) is a highly successful formal mathematical model of decision making under risk (Tversky & Kahneman, 1992). While CPT has been used extensively to describe typical patterns of decision making across the population, few studies have investigated the extent to which individuals differ in these decision-making patterns and how well CPT can account for these individual differences. However, recent advances in hierarchical Bayesian methods have improved the ability to estimate CPT parameters at the individual level (Nilsson, Rieskamp, & Wagenmakers, 2011). In the current study, we investigated individual differences in patterns of decision making under risk. Participants chose between sure outcomes and gambles. We focused on three key issues. First, we wanted to know whether participants' choices were best described as variations on a single pattern or whether there were multiple distinct patterns. We found multiple distinct patterns of choice behavior; indeed some participants were more likely to gamble in precisely the opposite conditions of other participants. Second, we evaluated the ability of a CPT-based model to account for this range of choice behavior. We established that, employing the fundamental CPT concept of a relative point of reference, a wide variety of decision-making behavior could be accounted for. And third, we investigated the ability of the CPT-based model to predict performance of the same participants on a second decision-making task using the parameter values from the initial task. We showed that, consistent with stable individual styles of decision making, the parameter values from a task with mixed gambles (i.e. gambles with both gains and losses as outcomes) were successful at predicting choices on a task with only positive gambles (i.e. gambles where all outcomes were gains). Overall, we found a diverse set of choice behaviors that were, nonetheless, well accounted for by CPT.

Keywords: decision making, uncertainty, risk, individual differences, choice behavior

1 Introduction

According to cumulative prospect theory (CPT), when faced with decisions involving uncertain but well-defined outcomes, known as decision making under risk, individuals subjectively evaluate the values and probabilities of each available option [4]. CPT has been highly successful as a description of choice behavior as it occurs in everyday life and across a variety of decision-making tasks. However, most studies have focused on the general pattern across participants within a single task, which may mask important differences in decision-making styles between participants and across tasks. To investigate these potential differences, the present study used recent advances in the ability to fit CPT-based models at the individual level using hierarchical Bayesian methods [3].

We studied decisions between two options: a sure outcome and a gamble with two probabilistic outcomes. Information about the final outcomes of the gambles was not revealed to participants until after they had completed the two tasks. This was done so that we could focus on participants' existing choice preferences without emphasizing trial-by-trial learning effects that would arise if outcomes were revealed after each trial. We believe it is important to characterize decision making in this way as a point of reference on which to then build theories that incorporate learning.

We focused on three key issues. First, we wanted to know whether participants' choices were best described as variations on a single pattern or whether there were multiple distinct patterns. Second, we evaluated the ability of a CPT-based model to account for the observed range of choice behavior across participants. And third, we investigated the ability of the CPT-based model to predict performance of the same participants on a second decision-making task using the parameter values from the initial task.

1.1 Model Definition

We used a CPT-based model with four parameters: probability distortion, γ, diminishing marginal utility, α, loss aversion, λ, and choice consistency, φ [1, 3]. This choice of parameterization strikes a good balance between flexibility and predictive value while being amenable to estimation [1, 3]. In this model, the decision-maker determines a subjective expected utility, EU, for an option, O, by summing across the n possible outcomes of that option as follows:

$$EU(O) = \sum_{i=1}^{n} w(p_i)v(x_i),$$

where each possible outcome has a probability, p, and a value, x. The objective probabilities are transformed into subjective decision weights by a probability function, $w()$, and the objective values are transformed into subjective utilities by a value function, $v()$. The probability function is defined as:

$$w(p) = \frac{p^{\gamma}}{(p^{\gamma}+(1-p)^{\gamma})^{1/\gamma}},$$

where a probability distortion parameter, $0 \leq \gamma \leq 1$, determines the degree of overweighting of small probabilities and underweighting of large probabilities (see Figure 2c). The value function is defined as:

$$v(x) = \begin{cases} x^{\alpha} & \text{if } x \geq 0 \\ -\lambda(-x)^{\alpha} & \text{if } x < 0 \end{cases},$$

where a diminishing marginal utility parameter, $0 \leq \alpha \leq 1$, determines the discounting of large magnitude values, and a loss aversion parameter, $0 \leq \lambda \leq 5$, determines the overweighting of losses compared to gains (see Figure 2c). Finally, the probability, $p()$, of choosing an option, A, over an alternative option, B, is determined from their subjective expected utilities according to a version of Luce's choice rule:

$$p(A,B) = \frac{1}{1 + e^{\varphi[EU(B)-EU(A)]}}$$

where a consistency parameter, $0 \leq \varphi \leq 5$, determines how much more likely it is that the option with the larger subjective expected value is chosen (see Figure 2c).

2 Methods

Two decision-making tasks consisting of decisions between a gamble and a sure outcome were completed by 68 healthy undergraduates. The gambles were presented as pie charts showing two monetary values with complementary probabilities (see Figure 1a and 1b). The two tasks were administered in separate sessions. Participants were informed that they would receive a monetary bonus of between $1 and $5 based on the selections they made after they completed both tasks. The positive task involved only positive monetary values while the mixed task involved both negative and positive values. In all trials on the mixed task, the gamble loss was a negative value, while the gamble win and the sure outcome were positive.

In both tasks, trials differed in the probability of winning the gamble, the magnitude of the winning gamble outcome, and whether the gamble or the sure outcome was the better choice. Each of these factors (Probability, Magnitude, and Better Choice) had two levels, for a total of eight trial types. The better outcome was either the gamble (G) with an EV of $5 more than the sure outcome, or the sure outcome (S) with an EV of $5 more than the gamble. The probability of winning the gamble was either high (H) (.55 to .75) or low (L) (.25 to .45). And the magnitude of the winning gamble outcome was either large (L) (positive task: $80 to $100; mixed task: $110 to $130) or small (S) (positive task: $40 to $70; mixed task: $80 to $100). The value of the losing gamble outcome in the positive task ranged from $10 to $25 and in the mixed task from −$10 to −$25. All values were multiples of $5. Each task had 128 task trials, with 16 in each of the 8 conditions. In order to identify participants not properly engaged with the task, we also included catch trials where one option dominated the other. Catch trials were not included in the main analysis.

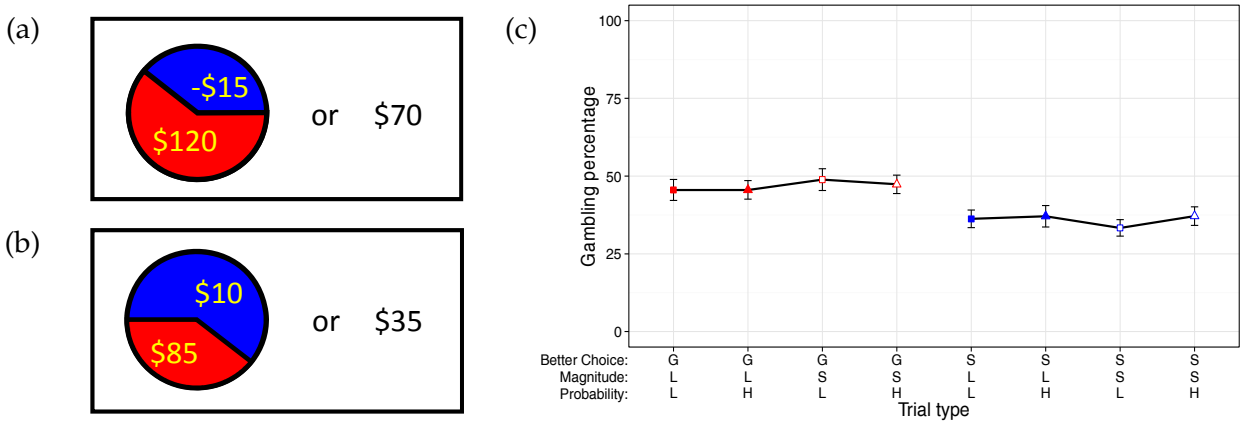


Figure 1. Examples of decisions in the (a) mixed and (b) positive tasks. (c) Mixed task results across all participants showing how often participants chose the gamble over the sure outcome for each of the different trial types.

2.1 Model Estimation

We adapted a hierarchical Bayesian Markov chain Monte Carlo (MCMC) parameter estimation method running in R and JAGS [3]. For each of the four model parameters, we estimated the mean and variance of the population distribution as well as individual parameter values. Uninformed priors were used for all parameters. A total of 30,000 samples were obtained across three chains after a burn-in of 1,000 samples per chain. The model was fit to the mixed task data, and we used the mean parameter values of the posterior distributions found for each participant to simulate their behavioral performance on the same trials the participants had completed in the lab, for both the mixed and positive tasks.

We fit two versions of the model with different reference points. For the "unshifted" version, $0 was used as the reference point for the values in each trial. In other words, the stated values were used as is. For the "shifted" model, the sure outcome was used as the reference point for values in each trial. The intuition

being that the participant knows they can pick the sure outcome, so the possible gamble outcomes are evaluated relative to the guaranteed sure gain.

3 Results

Participants had an average age of 20.6 years and were mostly female (79%). A total of seven particiants were dropped from final analysis due to high no-response rates and/or inadequate performance on catch trials, leaving 61 participants in the analysis. Aggregating results across all participants for the mixed task revealed that participants gambled more often when the gamble was the better choice than when the sure outcome was the better choice, ES = 10.88%, $F(1,60) = 51.27$, $p < .001$, but probability and magnitude did not have a significant influence (see Figure 1c).

Based on our *a priori* expectation from an earlier study, we used partitioning around medioids [2] to find four clusters of participants from gambling percentages with each of the eight trial types (see Figure 2a and 2b). A Duda-Hart test established that the participants grouped into multiple clusters, $p < .001$. These clusters demonstrated four distinct patterns of decision-making. They differ in the magnitude and direction of influence of the manipulations of probability, magnitude, and better choice on the percentage of the time they chose to gamble. The first cluster (Mixed: A) was most sensitive to expected value, gambling more often when the gamble had a higher EV than when the sure outcome had a higher EV, ES = 17.78%, $F(1, 16) = 37.08$, $p < .001$. The second cluster (Mixed: B) was most sensitive to probability, gambling more far often when the probability of winning the gamble was high than when it was low, ES = 62.45%, $F(1, 7) = 123.34$, $p < .001$. The third cluster (Mixed: C) was also most sensitive to probability, but they showed the opposite tendency, gambling more often when the probability was low than when it was high, ES = 36.72%, $F(1, 15) = 41.35$, $p < .001$. Finally, the distinguishing characteristic of the fourth cluster (Mixed: D) was their infrequency of gambling across all conditions.

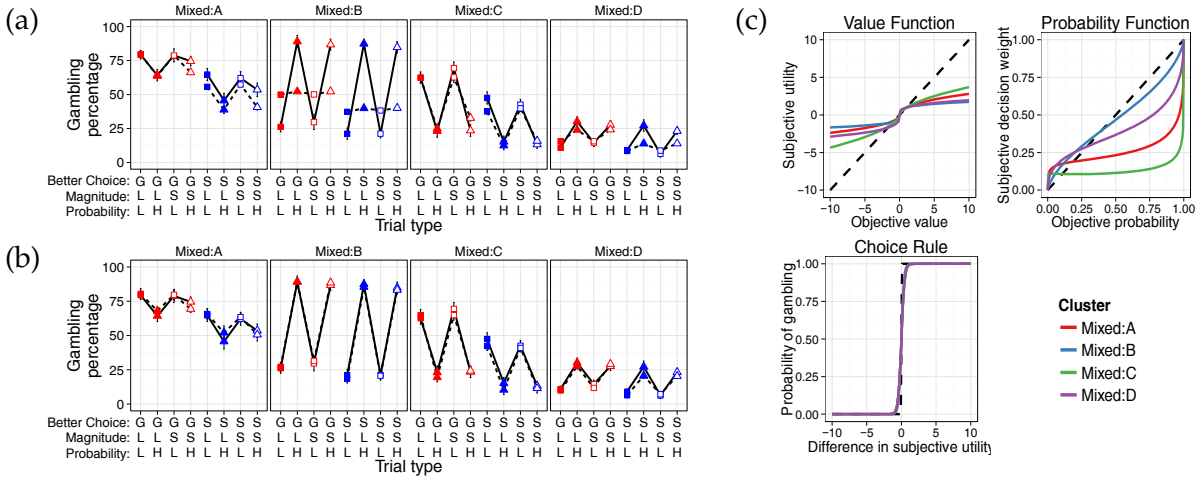


Figure 2. Mixed task results as a function of trial type for each of the behaviorally-defined clusters (solid lines) along with model fits based on (a) unshifted and (b) shifted reference points (dotted lines). (c) Value, probability, and choice functions for each cluster based on mean estimated parameter values for the shifted reference point model.

We then fit our unshifted and shifted reference point models (see Figure 2a and 2b). Diagnostic plots showed MCMC chain convergence for each parameter. Evaluating model fit at the level of individual participants, the shifted model, $R^2 = 0.91$, RMSE = 9.35, out-performed the unshifted model, $R^2 = 0.69$, RMSE = 17.25. Likewise, when fit was evaluated at the cluster level, the shifted model, $R^2 = 0.99$, RMSE = 3.01, out-performed the unshifted model, $R^2 = 0.60$, RMSE = 17.07. And finally, in terms of the deviance information criterion (DIC), the shifted model, DIC = 7545, is preferred to the unshifted model, DIC = 8266. The value function, probability function, and choice rule calculated by averaging the individual mean parameter values from each cluster for the preferred shifted reference point model are shown in Figure 2c.

Next we used participants' estimated parameter values from the shifted reference point model fit to the mixed task to predict choice behavior on the positive task. Model fit was evaluated at the individual participant level, $R^2 = 0.17$, RMSE = 26.09, and at the cluster level, $R^2 = 0.81$, RMSE = 10.63 (see Figure 3). To further assess the consistency of decision-making between tasks we fit the shifted reference point model directly to the positive task results, and then correlated the individual mean parameter values derived from the positive task with those from the mixed task. Values for three of the four parameters were correlated across tasks: α, $r(59) = .68$, $p < .001$; λ, $r(59) = .54$, $p < .001$; γ, $r(59) = .33$, $p = .008$; φ, $r(59) = .23$, $p = .07$.

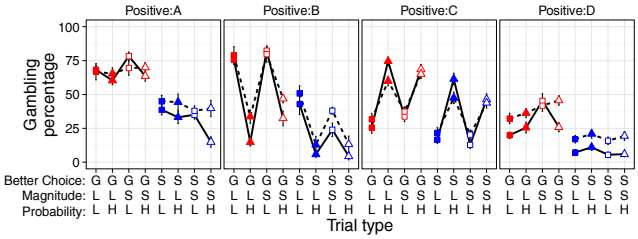


Figure 3. Positive task results as a function of trial type for each of the behaviorally defined clusters (solid lines), and predictions from the shifted model fit to the mixed task results (dotted lines).

4 Discussion

The present study reinforces the importance of looking beyond overall trends when attempting to characterize decision making. The aggregate results showed no significant effects of the manipulations of probability or magnitude on choice behavior, but a cluster analysis revealed that this was due to distinct patterns of performance that cancelled each other out in the overall results. In our tasks, some participants were quite sensitive to small differences in expected value. Other participants were most strongly influenced by manipulations of probability, with some gambling more often when they had a high probability of winning a little more, and others gambling more often when they had a small probability of winning a lot more. And still others tended not to gamble much no matter the options.

We probed the extent to which CPT would be useful for characterizing, not just the average trend, but the full range of individual results. We found that a CPT-based model employing the sure outcome as a relative reference point was able to account for the qualitatively different patterns of choice behavior through quantitative adjustments of model parameters. The use of a relative point of reference is consistent with the framing effects which were central to the development of CPT [4].

Finally, we were able to apply the individual parameter estimates from one task to predict performance on a second, albeit similar, task. This supports other findings that decision-making styles, characterized as CPT parameter values, can remain consistent across tasks and represent trait-like attributes of individuals [1]. In ongoing work, we are expanding the types of decisions being modeled in order to determine how stable these parameter values are, investigating their neural correlates using fMRI, and incorporating ambiguity and learning from outcomes into the model.

References

[1] Glöckner, A., & Pachur, T. (2012). Cognitive models of risky choice: parameter stability and predictive accuracy of prospect theory. *Cognition*, *123*(1), 21–32.

[2] Kaufman, L. & Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis.* New York: Wiley.

[3] Nilsson, H., Rieskamp, J., & Wagenmakers, E.-J. (2011). Hierarchical Bayesian parameter estimation for cumulative prospect theory. *Journal of Mathematical Psychology*, *55*(1), 84–93.

[4] Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, *5*(4), 297–323.

Learning for Multiagent Decentralized Control in Large Partially Observable Stochastic Environments

Miao Liu
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139
miaoliu@mit.edu

Christopher Amato
Department of Computer Science
University of New Hampshire
Durham, NH 03824
camato@cs.unh.edu

Emily P. Anesta
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420
eanesta@ll.mit.edu

J. Daniel Griffith
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420
dan.griffith@ll.mit.edu

Jonathan P. How
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
jhow@mit.edu

Abstract

This paper presents a probabilistic framework for learning decentralized control policies for cooperative multiagent systems operating in a large partially observable stochastic environment based on batch data (trajectories). In decentralized domains, because of communication limitations, the agents cannot share their entire belief states, so execution must proceed based on local information. Decentralized partially observable Markov decision processes (Dec-POMDPs) provide a general framework for modeling multiagent sequential decision making processes in the presence of uncertainty. Although Dec-POMDPs are typically intractable to solve for real-world problems, recent research on macro-actions in Dec-POMDPs has significantly increased the size of problems that can be solved. However, existing methods are confined to tree-based policies in finite-horizon problems, and assume the underlying POMDP models are known a priori. To accommodate more realistic scenarios when the full POMDP model is unavailable and the planning horizon is unbounded, this paper presents a policy-based reinforcement learning approach to learn the macro-action policies represented by Mealy machines. Based on trajectories of macro-actions, observations, and rewards generated by interacting with the environment with hand-coded policies (demonstrations) and random exploration, an expectation-maximization (EM) algorithm is proposed to learn the decentralized macro-action policies, leading to a new framework called POEM (Policy-based EM), which has convergence guarantee for bath learning. The performance of POEM is demonstrated on two domains, including a benchmark navigation-among-movable-obstacle problem, and a newly designed large search and rescue problem. Our empirical study shows POEM is a scalable batch learning method that can learn optimal policies and achieve policy improvement over hand-coded (suboptimal) policies for missions in partially observable stochastic environments.

Keywords: Dec-POMDPs, Reinforcement Learning, Multiagent Planning, Mealy Machine, Monte-Carlo Methods

Acknowledgements

This work was supported in part by the ONR under MURI program award #N000141110688. The Lincoln Laboratory portion of this work was sponsored by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States government.

1 Introduction

Multi-agent and multi-robot systems are becoming an important solution to many real-world problems. For example, in disaster situations, such as earthquake, hurricane, or terrorist attacks, it is urgent that trapped survivors can be found and rescued within 48 hours. Otherwise, the chance of finding victims alive decreases substantially [7]. To solve a search and rescue (SAR) problem efficiently, a team of ground vehicles and aerial vehicles have to be deployed to locate, rescue and medically stabilize survivors trapped in hazardous spaces. This SAR problem in its most general form can be formulated as a decentralized partially observable Markov decision process (Dec-POMDP) [2, 12], where a team of agents must cooperate to optimize some global objective in the presence of uncertainty. Moreover, because of possibly limited range of communication, each agent has to make its own decisions based on its own local observations when there are no other agents in the effective communication range. To date, researches have achieved significant progress on solving the cooperative multiagent sequential decision-making problems that arise in numerous applications including transportation [3], extra-planetary exploration [5], and traffic control [14]. However, most Dec-POMDP formulations assume low level state-action granularity and operate with primitive actions which last exactly one time step, so large problems remain intractable.

Recent research has addressed the more scalable macro-action based Dec-POMDP (MacDec-POMDP) case where each agent has macro-actions (temporally extended actions), which may require different amounts of time to execute [4]. However, current MacDec-POMDP methods [4] require knowing domain models a priori, and are confined to tree-based policy representations in finite horizon problems. For many real-world problems, such as SAR, the exact domain model may not be directly available, and the planning horizon might be indefinite. To solve long horizon (or infinite-horizon) MacDec-POMDP problems when the domain models are unknown, we propose a policy-based expectation maximization (POEM) algorithm to learn the macro-action based finite-state controllers (FSCs). POEM adopts a special type of FSC, Mealy machine [1] for policy representations, and performs batch off-policy reinforcement learning (RL) based on trajectories collected by executing hand-coded and exploration policies.

2 Background

A **Dec-POMDP** can be represented as a tuple $\langle \mathcal{N}, \mathcal{A}, \mathcal{S}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, \gamma \rangle$, where $\mathcal{N}$ is a finite set of agent indices; $\mathcal{A} = \otimes_n \mathcal{A}_n$ and $\mathcal{O} = \otimes_n \mathcal{O}_n$ respectively are sets of joint actions and observations, with $\mathcal{A}_n$ and $\mathcal{O}_n$ available to agent n. At each step, action $\vec{a} = (a_1, \cdots, a_N) \in \mathcal{A}$ is selected and a joint observation $\vec{o} = (o_1, \cdots, o_N)$ is received; $\mathcal{S}$ is a set of finite world states; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function with $\mathcal{T}(s'|s, \vec{a})$ denoting the probability of transitioning to s' after taking joint action $\vec{a}$ in s; $\Omega : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ is the observation function with $\Omega(\vec{o}|s', \vec{a})$ the probability of observing $\vec{o}$ after taking joint action $\vec{a}$ and arriving in state s'; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function with $r(s, \vec{a})$ the immediate reward received after taking joint action $\vec{a}$ in s; $\gamma \in [0, 1)$ is a discount factor. Because each agent lacks access to other agents' observations, each agent maintains a local policy Ψ_n, defined as a mapping from local observation histories to actions. A joint policy consists of the local policies of all agents. For an infinite-horizon Dec-POMDP with initial belief state b_0, the objective is to find a joint policy $\Psi = \otimes_n \Psi_n$, such that the value of Ψ starting from b_0, $V^{\Psi}(b(s_0)) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \vec{a}_t)|b_0, \Psi\big]$, is maximized.

A **MacDec-POMDP** with local macro-action (MA) is defined as a Dec-POMDP where $\mathcal{M}_n$ represents a finite set of MA for each agent n, with $\mathcal{M} = \otimes \mathcal{M}_n$ the set of joint MAs. Here, MAs are represented by options [13]. Given $\mathcal{H}_n$, the observation history of agent n's local option is defined by the tuple: $\mathcal{M}_n = \langle \beta_n^m, \mathcal{I}_n^m, \pi_n^m \rangle$, where $\beta_n^m : \mathcal{H}_n \to [0, 1]$ is a stochastic termination condition, $\mathcal{I}_n^m \in \mathcal{H}_n$ is the initiation set and $\pi_n^m : \mathcal{H}_n \times \mathcal{A}_n \to [0, 1]$ is an option policy for macro-action m. Macro-actions are natural representations for robot or human operation for completing a task (e.g., navigating to a way point or placing an object on a robot).

A **FSC** is a compact way to represent a policy as a mapping from histories to actions. Formally, a stochastic FSC for agent n is defined as a tuple $\Theta_n = \langle \mathcal{A}_n, \mathcal{O}_n, \mathcal{Z}_n, \mu_n, W_n, \pi_n \rangle$, where, $\mathcal{A}_n$ and $\mathcal{O}_n$ are the same as defined in the Dec-POMDP; $\mathcal{Z}_n$ is a finite set of nodes used to represent the policy for agent n; μ_n is the initial node distribution with μ_n^z the probability of agent n initially being in z; W_n is a set of Markov transition matrices with $W_{n,o}^{z,z'}$ denoting the probability of the controller node transiting from z to z' when agent n in z sees observation o; π_n is a set of stochastic policies with $\pi_{n,z}^{a,o}$ the probability of agent n taking action a in z after seeing observation o. This type of FSC is called Mealy machine [1], where an agent's local policy for action selection $\pi_{n,z}^{a,o}$ depends on both current controller node (an abstraction of history) and immediate observation. There is another type of FSCs called Moore machines, where π_n^a only depends on controller node, is also widely used for (Dec-)POMDP policy representations. In contrast to a Moore machine, by conditioning action selections on additional immediate observations, a Mealy machine can use this observable information to initiate a valid macro-action controllers when the MacDec-POMDP model is unknown.

There are several features in MacDec-POMDP frameworks that benefit policy learning. Firstly, observations are determined by the termination condition of macro-actions (although a more general model of high-level observations can be included). Therefore, the observation set can be encoded as $\bar{\Omega} = \{Null, \Omega\}$, where Ω is a set indicating what agents see after the corresponding macro-action termination condition is satisfied, and $Null$ indicates the same macro-action is still in execution. Additionally, controllers stay in the same nodes until current macro-actions are completed, hence the diagonal elements of W corresponding to each $o \in \Omega$ can be set to zero, i.e. $W(z_\tau = z|z_{\tau-1} = z, o_\tau \neq Null) = 0$. Moreover, before a macro-action terminates, the agent stays in the same controller node, i.e. $W(z_\tau = z|z_{\tau-1} = z, o_\tau = Null) = 1$.

A Dec-POMDP planning problem can be transformed into an **inference problem** and then efficiently solved by an EM algorithm. Previous EM methods [8] have achieved success in scaling to larger problems, but these methods require using a Dec-POMDP model

both to construct a Bayes net and to evaluate policies. When the exact model parameters $\mathcal{T}$, Ω and $\mathcal{R}$ are unknown, an RL problem must be solved instead. To address this important yet less addressed problem in decentralized domains, a global empirical value function is constructed based on the macro-action, observation and reward trajectories, and the product of local policies of all agents.

Definition 1. *(Global empirical value function) Let* $\mathcal{D}^{(K)}=\{(\vec{o}_0^k\vec{m}_0^k r_0^k\cdots\vec{o}_{T_k}^k\vec{a}_{T_k}^k\ r_{T_k}^k)\}_{k=1}^K$ *be a set of episodes resulting from* N *agents who choose actions according to* $\Psi=\otimes_n\Psi_n$*, a set of arbitrary stochastic policies with* $p^{\Psi_n}(a|h)>0$*,* $\forall$ *action* a*,* $\forall$ *history* h*. The global empirical value function is defined as* $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)\stackrel{def.}{=}\sum_{k=1}^K\sum_{t=0}^{T_k}\frac{\gamma^t(r_t^k-R_{min})\prod_{\tau=0}^t\prod_{n=1}^N p(m_{n,\tau}^k|h_{n,\tau}^k,\Theta_n)}{\prod_{\tau=0}^t\prod_{n=1}^N p^{\Psi_n}(m_{n,\tau}^k|h_{n,\tau}^k)}$ *where* $h_{n,t}^k=(m_{n,0:t-1}^k,o_{n,1:t}^k)$*,* $0\le\gamma<1$ *is the discount and* R_{min} *is the minimum reward.*

Definition 1 provides an off-policy learning objective: given data $\mathcal{D}^{(K)}$ generated from a set of behavior policies Ψ, find a set of decentralized policies $\Theta=\{\Theta_i\}_{i=1}^N$ such that $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ is maximized. Here, we assume factorized policy representation $p(\vec{m}_{0:\tau}^k|\vec{h}_{1:\tau},\Theta)=\prod_{n=1}^N p(m_{n,\tau}^k|h_{n,\tau}^k,\Theta_n)$ to accommodate decentralized policy execution.

3 MacDec-POMDP Policy Learning by Expectation Maximization

Direct maximization of $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ is difficult; instead, we maximize the lower bound of the logarithm of $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$. To this end, we apply Jensen's inequality to obtain obtain a lower bound of $\ln\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ augmented by node sequences $\{\vec{z}_t^k\}$:

$$\ln\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)=\ln\sum\nolimits_{k,t,\vec{z}_{0:t}^k}\frac{\tilde{r}_t^k\,q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})}{K}\,\frac{p(\vec{m}_{0:t}^k,\vec{z}_{0:t}^k|\vec{o}_{1:t},\Theta)}{q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})}\ge\sum\nolimits_{k,t,\vec{z}_t^k}\frac{q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})}{K}\ln\frac{\tilde{r}_t^k\,p(\vec{m}_{0:t}^k,\vec{z}_{0:t}^k|\vec{o}_{1:t}^k,\Theta)}{q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})}\stackrel{def.}{=}\mathrm{lb}(\Theta|\widetilde{\Theta}),\ (1)$$

where $\tilde{r}_t^k=\gamma^t r_t^k/\prod_{\tau=0}^t p^{\Psi}(\vec{m}_\tau^k|h_\tau^k),\forall t,k$ are reweighted rewards, and $\{q(\vec{z}_{0:t}^k|\widetilde{\Theta})\ge 0\}$ satisfy the normalization constraint $\sum_{k=1}^K\sum_{t=0}^{T_k}\sum_{\vec{z}_{0:t}^k}q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})=K$ with $\widetilde{\Theta}$ the most recent estimate of Θ.

Theorem 2. *Define* $\mathcal{F}=\Big\{\Theta=\{\Theta_n\}_{n=1}^N$ *with* $\Theta_n=(\mu_n,\pi_n,W_n):\sum_{u_n=1}^{|\mathcal{Z}_n|}\mu_n^{u_n}=1,\sum_{m_n=1}^{|\mathcal{M}_n|}\pi_{n,u_n}^{m_n,o_n}=1,\sum_{u_n=1}^{|\mathcal{Z}_n|}W_{n,o_n}^{v_n,u_n}=1,v_n=1\cdots|\mathcal{Z}_n|,\ m_n=1\cdots|\mathcal{M}_n|,\ o_n=1\cdots|\mathcal{O}_n|\Big\}$*, and* $\Theta^{(m)}$ *be a sequence produced by the iterative update* $\Theta^{(m+1)}=\arg\max_{\Theta\in\mathcal{F}}\mathrm{lb}(\Theta|\Theta^{(m)})$*, where* $\Theta^{(0)}$ *is an arbitrary initialization, then* $\{\Theta^{(m)}\}_{m\ge0}$ *monotonically increases (1), until convergence to a maxima.*

Algorithm 1 POEM

Require: Episodes $\mathcal{D}^{(K)}$ and the number of agents n,
1: **while** $\Delta\mathrm{lb}<\epsilon$ **do**
2: **for** $k=1$ to K, $n=1$ to N **do**
3: Compute action selection prob. $p(m_{n,0:t}^k|o_{n,1:t}^k,\widetilde{\Theta}),\forall t$, and forward-backward variables $\alpha_{n,k}$ and $\beta_{n,k}$
4: **end for**
5: Compute lb using $\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)=\frac{1}{K}\sum_{k=1}^K\sum_{t=1}^{T_k}\sigma_t^k\big(\widetilde{\Theta}\big)$
6: **for** $n=1$ to N **do**
7: Compute the ξ and ϕ variables, update Θ_n using (2)
8: **end for**
9: **end while**
10: **return** Parameters of the MacDec-POMDP policy, $\{\Theta_n\}_{n=1}^N$

Given the empirical value function in Definition 1 and its lower bound (1), the POEM algorithm is derived to learn the macro-action FSCs. The POEM algorithm is guaranteed to monotonically increase the empirical value function over successive iterations and converges to a local maximum. The convergence property is summarized by theorem 2. The proof for Theorem. 2 is an extension of the single agent case [9], which is omitted here because of space limitations. Algorithmically, the main steps of POEM involve alternating between computing the lower bound of the log empirical value function (E-step) and parameter estimation (M-step). The complete algorithm is summarized in Algorithm 1, and computational details are discussed next.

Computation of Lower Bounds (E-step) Theorem 2 and inequality (1) yield the lower bound $\mathrm{lb}\big(\Theta|\widetilde{\Theta}\big)\ge\ln\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$, where $\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$ is the value of the policy parameterized by $\widetilde{\Theta}$ (computed on line 5 in Algorithm 1). Computing $\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$ is equivalent to policy-evaluation, which uses all available episodes with the rewards are reweighted by the action selection probability $p(m_{n,0:t}^k|h_{o,0:t}^k,\widetilde{\Theta})$ to reflect the improved value of the new policy updated in the previous M-step. Note that $\mathrm{lb}(\Theta|\widetilde{\Theta})$ in (1) is maximized when $q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})=\tilde{r}_t^k p(\vec{m}_{0:t}^k,\vec{z}_{0:t}^k|\vec{o}_{1:t}^k,\widetilde{\Theta})/\hat{V}(\mathcal{D}^{(K)};\widetilde{\Theta})$, which is equal to $\tilde{r}_t^k p(\vec{a}_{0:t}^k|\vec{o}_{1:t}^k,\widetilde{\Theta})p(\vec{z}_{0:t}^k|\vec{m}_{0:t}^k,\vec{o}_{1:t}^k,\widetilde{\Theta})/\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$ where $p(\vec{z}_{0:t}^k|\vec{m}_{0:t}^k,\vec{o}_{1:t}^k,\widetilde{\Theta})$ is the joint distribution of controller nodes for all agents, and $\sigma_t^k(\widetilde{\Theta})\stackrel{\mathrm{def.}}{=}\tilde{r}_t^k p\big(\vec{m}_{0:t}^k|\vec{o}_{1:t}^k,\widetilde{\Theta}\big)=\Pi_{n=1}^N\tilde{r}_t^k p\big(m_{n,0:t}^k|o_{n,1:t}^k,\widetilde{\Theta}\big)$ is a reweighted reward.

Update of Policy Parameters (M-step) After computing the reweighted rewards $\{\sigma_t^k\}$ and the posterior distribution of controller nodes $p(\vec{z}_{0:t}^k|\vec{m}_{0:t}^k,\vec{o}_{1:t}^k,\widetilde{\Theta}),\forall t,\forall k$, the policy parameters is updated by $\Theta=\arg\max_{\Theta\in\mathcal{F}}\mathrm{lb}(\Theta|\widetilde{\Theta})$, subject to normalization constraints. Specifically, let $\xi_{n,k}^{t,\tau}(u_n,v_n)\stackrel{def.}{=}p(z_{n,\tau}^k=u_n,z_{n,\tau+1}^k=v_n|m_{n,0:t}^k,o_{n,1:t}^k,\widetilde{\Theta}_n)$ and $\phi_{n,k}^{t,\tau}(u_n)\stackrel{def.}{=}p(z_{n,\tau}^k=u_n|m_{n,0:t}^k,o_{n,1:t}^k,\widetilde{\Theta}_n)$. Expanding the lower bound of $\ln\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ and keeping the terms related to Θ, we have

$$\mathrm{lb}\big(\Theta|\widetilde{\Theta}\big)\propto\sum\nolimits_{k,t}\sigma_t^k(\widetilde{\Theta})\sum\nolimits_{n=1}^N\Big\{\sum\nolimits_{u_n=1}^{|\mathcal{Z}_n|}\phi_{n,k}^{t,0}(u_n)\ln\mu_n^{u_n}+\sum\nolimits_{\tau=0}^t\big[\sum\nolimits_{v_n=1}^{|\mathcal{Z}_n|}\phi_{t,\tau}^{n,k}(u_n)\ln\pi_{n,u_n}^{a_{n,\tau}^k,o_{n,\tau}^k}+\sum\nolimits_{u_n,v_n=1}^{|\mathcal{Z}_n|}\xi_{n,k}^{t,\tau}(u_n,v_n)\ln W_{n,o_{n,\tau+1}^k}^{u_n,v_n}\big]\Big\}.$$

Therefore, an analytic solution to problem $\Theta = \arg\max_{\Theta \in \mathcal{F}} \mathrm{lb}(\Theta|\widetilde{\Theta})$ can be obtained as

$$W_{n,o_n}^{u_n,v_n} \propto \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{\tau=0}^{t} \sigma_t^k \xi_{n,k}^{t,\tau}(u_n, v_n) \mathbb{I}(o_{n,\tau+1}^k = o_n), \forall n \in \mathcal{N}, u_n, v_n \in \mathcal{Z}_n, a_n \in \mathcal{A}_n, o_n \in \mathcal{O}_n. \quad (2)$$

where is $\mathbb{I}(\cdot)$ is the indicator function. π, μ are updated in similar ways. These updates constitute a policy-improvement step where the reweighted rewards are used to further improve policies.

Both the above steps require $\xi_{n,k}^{t,\tau}(u_n, v_n)$, which are computed based on $\alpha_{n,k}^{\tau} = p\big(z_{n,\tau}^k | m_{n,0:\tau}^k, o_{n,1:\tau}^k, \widetilde{\Theta}_n\big)$ and $\beta_{n,k}^{t,\tau} = \frac{p(a_{n,\tau+1:t}^k | z_{n,\tau}^k, o_{n,\tau+1:t}^k, \widetilde{\Theta}_n)}{\prod_{\tau'=\tau}^{t} p(m_{\tau}^k | h_{n,\tau'}^k, \widetilde{\Theta}_n)}$, $\forall n, k, t, \tau$. The (α, β) are forward-backward messages.

4 Experiments

We evaluate the performance of Algorithm 1 on both a benchmark domain of robot navigation and a large domain motivated by SAR.

4.1 A Navigation Among Movable Obstacles (NAMO) Problem

We first consider the NAMO problem, introduced in [4]. Here as shown in Figure 1, both agents are trying to reach a goal location "G" and have to move obstacles in the way. The primitive actions for each robot include move in four directions (up, down, left and right) and a "push" action to attempt to move a box to a specific location. The push action fails and the robot stays in place when the robot is not in front of the box. The small boxes (b_1 and b_2) can be moved by a single robot, and the large box (b_3) requires two robots to push together. Observations are an agent's own location and whether the large box or the same numbered box has been moved. There is noise in both navigation and in box movement: movement is successful with probability 0.9 and pushing the small and large boxes is successful with probability 0.9 and 0.8, respectively. To encourage the robots to reach the goal as quickly as possible, there is a negative reward (-1) when any agent is not in the goal region.

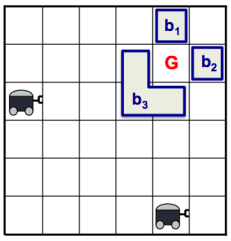


Figure 1: A 6×6 NAMO problem.

There were four macro-actions (option) defined for each agent, including 1) moving to a designated location to push the big box, 2) attempting to push the large box, 3) pushing the designated small box to the corner square, and 4) moving to the goal. The option of moving to the goal is only valid when at least one box has been moved and movement of any box is only valid if the large box and agent's designed box has not yet been moved. Movement options and pushing options terminate with the box successfully or unsuccessfully moved. These options provide high-level choices for the agents to coordinate on this problem, while abstracting away the navigation tasks to option execution.

Here, we want to examine the effect of the behavior policy Π on Algorithm 1's performance. A semi-random policy is applied to collect samples. Specifically, the learning agent is allowed access to episodes collected by taking actions according to a MaDec-POMDP algorithm (Option-based dynamic programming (O-DP)) [4]. Let $\eta\%$ be the probability that the agents follow the O-DP policy and $1 - \eta$ be the probability that the agents take random actions. The use of an O-DP policy is similar to the meta-queries used in [6], where a meta-query consults a domain expert for the optimal action at a particular time step. For each run of algorithm 1, $K = 100$ episodes of 10 steps are used to learn the FSCs with $|\mathcal{Z}_n| = 20, \forall n \in \mathcal{N}$, and the learned policies are evaluated by the discounted ($\gamma = 0.9$) accumulated reward averaged over 100 test episodes of 1000 steps. The results with $\eta = [0, 25, 50, 75, 100]$ are reported in Figure 2, which shows that with a small amount of expert knowledge ($\eta \geq 25\%$), our algorithm is able to recover the optimal policy.

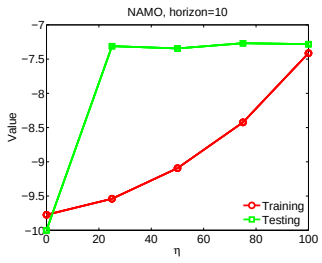


Figure 2: The performance of Algorithm 1 as a function of $\eta\%$ of data generated from an optimal policy.

4.2 A Search and Rescue (SAR) Problem

To further demonstrate the scalability and learning efficiency of the proposed algorithm, we designed a SAR problem involving four heterogeneous agents: an unmanned aerial vehicle (UAV) and three unmanned ground vehicles (UGVs). These agents operate in a 20×10 gridworld (shown in figure 3 (a)), where there are 6 rescue sites with different distance to the muster, which is the assembly area for the SAR agents. There are six victims, each of which is located in a different rescue site. There is a set of initial health states for each victim (not equal to each other) which do not change, though the initial position of each victim does change. Health trajectories are linear. There is a 5% noise in the observations, communication, and UAV's path. The speed of UAV is three times faster than UGV. However, only the UGVs can pick-up victims. The agents receive a positive reward (+1), if a victim transported to the muster is still alive (i.e., has health greater than zero), and receive a negative reward (-1) when one victim dies. For a UGV, there are 1120 observations encoded by the set $\Omega_{UGV} = SL \times SS \times SV \times OL \times OV$, where $SL = \{$site 1,..., site 6, muster$\}$ is a set of self location; $SS = \{$holding victim, not holding victim$\}$ is a set of self states; $SV = \{$has victims needing help, no victims needing help, unknown, critical victims$\}$ is the set of states of the victim at an agent's

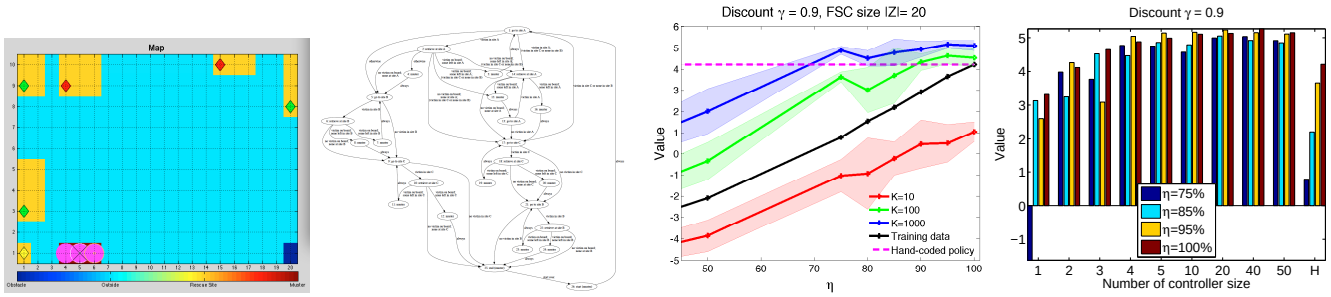


Figure 3: (a) A SAR domain (diamonds represent victims with colors indicate health, circles and cross represent UGV and UAV respectively); (b) A heuristic policy finite state machine for UGV constructed by domain experts; (c) Testing performance using different number of training sample K and percentage of hand-coded policies η; (d) Testing performance of policies with different FSC sizes.

current location; OV is the state of the victim at OL, the other location (from communication), and there are 8 macro-actions, including $\{m_1, \cdots, m_6\}$: go to one of the six sites, m_7: retrieve a victim at one site (retrieve) and m_8: go to muster and drop off victim (muster). For the UAV, there are 560 observations (assuming the UAV cannot hold victims) and 7 macro-actions, including go to muster and go to one of the six sites. All vehicles begin in the muster.

Given the large problem size, unknown POMDP model and the stochasticity in observations and communication, it is difficult to generate an optimal policy with existing solvers. Instead, a domain expert's knowledge is used to create a heuristic controller for exploration. Figure 3(b) provides a visualization of the graph describing the heuristic policy finite state machine. This is the policy for one ground vehicle, where sites A, B, and C represent a subset of the possible sites. The agent-to-site mappings include: a) agent 1: A=6, B=5, C=4; b) agent 2: A=3, B=1, C=2; c) agent 3: A=2, B=1, C=3. The policy for the air vehicle is to cycle linearly through the following sequence: site 1, site 2, muster, site 3, site 4, site 5, site 6, muster. The value of hand-coded policy is estimated based on 1000 runs with randomly placed victims and the mean reward is 4.22 (pink dotted line in Figure 3(c)).

To evaluate POEM's performance on the SAR domain, we test $\eta = [0, 25, 50, 75, 80, 85, 90, 95, 100]$. For each setting of η, 2000 training trajectories (with horizon upper bound set to 200) are generated. When setting $|\mathcal{Z}_n| = 20$ and $K = 1000$, the training time is less than 15min on average. The corresponding testing results are plotted in figure 3(c), from which we can see, a) POEM is able to achieve policy improvement with a sufficient amount of samples ($K > 100$); b) By using 1000 trajectories generated from heuristic policy, POEM is able to achieve a mean value greater than 5, which is higher than the value of hand-coded policy; c) Adding a small amount of noise (5%) to the heuristic policy can help getting a slightly better performance than purely using handed coded policy. Hence the experiments demonstrate POEM can achieve policy improvement and is scalable to large problems. In addition, nine settings of $|\mathcal{Z}_n|$ are used to investigate the influence of the controller size on policy quality. As shown in Figure 3(d), the FSCs learned by POEM render much higher value than the hand-coded policies (with x-axis labeled with H) over a wide range of choices for $|\mathcal{Z}_n|$ (from 5 to 50), indicating robustness of POEM to the size of controller nodes. However, when $|\mathcal{Z}_n|$ is too small, POEM cannot accommodate a good policy. Automatic inference of the necessary size of controllers can be performed using nonparametric methods, such as the hierarchical Dirichlet process [11] and stick-breaking processes [10], which is left for future work.

5 Conclusions

This paper presents an RL method for learning to coordinate multiple agents in macro-action level Dec-POMDPs, an important problem that has not been adequately addressed before. Our method uses previously executed macro-action, observation histories and rewards to improve future decision making without explicitly requiring mission models. An algorithm called POEM is presented for batch learning with convergence guarantee (local optimal). Theoretical analysis and empirical results show the proposed method is a promising tool for solving RL in MacDec-POMDPs.

References

[1] C. Amato, B. Bonet, and S. Zilberstein. Finite-state controllers based on mealy machines for centralized and decentralized pomdps. In *AAAI*, 2010.
[2] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer. Decentralized control of partially observable Markov decision processes. In *CDC*, 2013.
[3] C. Amato, G. D. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *ICRA*, 2015.
[4] C. Amato, G. D. Konidaris, and L. P. Kaelbling. Planning with macro-actions in decentralized pomdps. In *AAMAS*, 2014.
[5] D. Bernstein, S. Zilberstein, R. Washington, and J. Bresina. Planetary rover control as a Markov decision process. In *Int'l Symp. on AI, Robot. & Automation in Space*, 2001.
[6] F. Doshi-Velez, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. *Artificial Intelligence*, 187:115–132, 2012.
[7] S. Grayson. Search & Rescue using Multi-Robot Systems. `http://www.maths.tcd.ie/~graysons/documents/COMP47130_SurveyPaper.pdf`.
[8] A. Kumar and S. Zilberstein. Anytime planning for decentralized POMDPs using expectation. In *UAI*, pages 2140–2146, 2010.
[9] H. Li, X. Liao, and L. Carin. Multi-task reinforcement learning in partially observable stochastic environments. *JMLR*, 10:1131–1186, 2009.
[10] M. Liu, C. Amato, X. Liao, J. P. How, and L. Carin. Stick-Breaking Policy Learning in DEC-POMDPs. In *IJCAI (to appear)*, 2015.
[11] M. Liu, X. Liao, and L. C. and. Infinite regionalized policy representation. In *Proc. of the 28th Int'l Conf. on Machine Learning*, pages 769–776, 2011.
[12] F. A. Oliehoek. Decentralized POMDPs. In *Reinforcement Learning: State of the Art, Adaptation, Learning, and Optimization*, pages 471–503. Springer, 2012.
[13] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.
[14] F. Wu, S. Zilberstein, and N. R. Jennings. Monte-carlo expectation maximization for decentralized POMDPs. In *IJCAI*, pages 317–403, 2013.

Learning and Planning with Timing Information in Markov Decision Processes

Pierre-Luc Bacon, Borja Balle and Doina Precup
Reasoning and Learning Lab
McGill University
Montreal, Canada
{pbacon,bballe,dprecup}@cs.mcgill.ca

Abstract

We consider the problem of learning and planning in Markov decision processes with temporally extended actions represented in the options framework. We propose to use predictions about the duration of extended actions to represent the state and show that this leads to a compact predictive state representation model independent of the set of primitive actions. Then we develop a consistent and efficient spectral learning algorithm for such models. Using just the timing information to represent states allows for faster improvement in the planning performance. We illustrate our approach with experiments in both synthetic and robot navigation domains.

Keywords: Predictive State Representations, Options, Planning

Acknowledgements

This work was supported by the Fonds Quebecois de la Recherche sur la Nature et les Technologies (FQRNT) and the Natural Sciences and Engineering Research Council (NSERC)

1 Introduction

Modelling the dynamics of an agent embedded in a large, complex environment is key to building good planning algorithms for such agents. In most practical applications, models are carefully designed by hand, and the agent's "state" is given by measurements which are understandable by the designer of the system (such as spatial location and velocity, in the case of a robot). However, learning dynamical models for such states from data, as well as planning with them can be quite tricky. An alternative idea is to use models that are "subjective", centered on the agent's own perception and action capabilities. For example, affordances [Gibson, 1977] describe "state" through the courses of action that are enabled. Similarly, in robotics, subjective representations have been used to model dynamics, e.g. [Bowling *et al.*, 2005; Stober *et al.*, 2011]. Such models are appealing from a psychological point of view, but run into computational problems in very large observation spaces.

In this paper, we focus on a special class of subjective models, *timing models*, which arise from restricting the observations available to the agent to just information about the duration of certain courses of action. Timing of events is understood to be crucial to animal learning [Machado *et al.*, 2009]. The goal of this paper, however, is not learning of the timing of external events, but rather to learn the duration of courses of action that an agent might take. The ensemble of such durations will constitute the agent's *state*, which will be maintained as new data is received. We use the framework of options [Sutton *et al.*, 1999] to model extended courses of actions, and we present an approach for learning option durations.

Our models over durations can be viewed as affordances if we consider an option to be available if its estimated duration is within some reasonable bounds. Note that these models are much simpler than full option models, which provide joint information on the timing as well as the state or observation in which the option will terminate, e.g. [Wolfe and Singh, 2006]. Our approach can also be interpreted as a computationally and statistically efficient way of exploiting prior information about useful courses of action provided in the form of options. As a consequence, the size of our models is independent of the number of possible primitive actions in the underlying system. Another interesting feature of our approach is that we are able to learn feature representations for states using timing information only; this means our method can be applied to observable settings with high-dimensional observations and to partially observable settings as well.

Of course, the utility of such timing models depends strongly on the nature of the task to be solved by the agent, as well as on the "quality" of the options available to the agent. The simplest example in which option duration models are beneficial is that of minimum time to goal problems, in which an agent receives a fixed penalty per time step until its task is completed. In this case, knowing the duration of an option immediately gives us the reward model, so the option duration model has direct value for a planner. More generally, option duration models are beneficial as a form of localization. If you imagine a robot that has models for options that move radially out from the current position, this would allow localizing with respect to all neighboring walls. Finally, consider a problem in which a financial agent is holding stocks, and options which hold a particular stock while it is above a certain value, and sell under that value. In this case, timing models tell us exactly when stocks would be crossing certain barriers. It is clear in this case that, even though we are estimating only durations, these encode important state information (because of the way in which the options are defined).

In this paper we analyze the capacity of option duration models to represent states in a Markov Decision Process (MDP). We propose a spectral algorithm for learning option duration models which builds on existing work for learning transformed predictive state representations [Rosencrantz *et al.*, 2004a]. Finally we evaluate the quality of learning and planning with our model in experiments with discrete MDPs.

1.1 Markov Decision Processes and Temporally Extended Actions

A *Markov decision process* (MDP) is a tuple $M = \langle S, A, P, R \rangle$ where S is the state space, A is the action set, $P : S \times A \to (S \to [0, 1])$ defines a probability distribution over next states, and $R : S \times A \to \mathbb{R}$ is the expected reward function (see [Puterman, 1994] for a review). We refer to probability distributions on S by α, but sometimes use $\boldsymbol{\alpha}$ to stress that we view them as vectors in $\mathbb{R}^S$. Suppose α is a distribution over S and $\pi : S \times A \to [0, 1]$ is a stochastic action policy which, given state s, chooses action a with probability $\pi(s, a)$. The environment then returns a state sampled from P; and the resulting state distribution α' is given by:

$$\alpha'(s') = \sum_{s \in S} \alpha(s) \sum_{a \in A} \pi(s, a) P(s, a)(s') \ . \tag{1}$$

Temporal abstraction in MDPs has been used as a tool to speed up learning and planning algorithms. We adopt the framework of options [Sutton *et al.*, 1999], with the goal of learning state representations based on option timing models. An *option* is a tuple $\omega = \langle I_\omega, \pi_\omega, \beta_\omega \rangle$ where $I_\omega \subseteq S$ is the set of initial states, $\pi_\omega : S \times A \to [0, 1]$ is the option's stochastic action policy, and $\beta_\omega : S \to [0, 1]$ is the option termination probability for each state.

1.2 Predictive State Representations

A *predictive state representation* is a model of a dynamical system where the current state is represented as a set of predictions about the future behavior of the system [Littman *et al.*, 2002; Singh *et al.*, 2004]. We use a particular instantiation of this general idea, the so-called *transformed linear predictive state representation* [Rosencrantz *et al.*, 2004b], which we abbreviate for simplicity as PSR.

A PSR with observations in a finite set Σ is a tuple $\mathcal{A} = \langle \boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}_{\sigma\in\Sigma}\rangle$ where $\boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$ are the initial and final weights respectively, and $\mathbf{A}_\sigma \in \mathbb{R}^{n\times n}$ are the transition weights. The dimension n of these vectors and matrices is the *number of states* of the PSR. The function $f_\mathcal{A} : \Sigma^\star \to \mathbb{R}$ computed by $\mathcal{A}$ assigns a number to each string $x = x_1 x_2 \cdots x_t \in \Sigma^\star$ as follows:

$$f_\mathcal{A}(x) = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_{x_1}\mathbf{A}_{x_2}\cdots\mathbf{A}_{x_t}\boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty \ . \tag{2}$$

The behavior of a stochastic dynamical system producing observations in a finite set Σ can be entirely characterized by the function $f : \Sigma^\star \to \mathbb{R}$ giving the probability $f(x)$ of observing each possible sequence of observations x. A convenient algebraic way to summarize all the information conveyed by f is its *Hankel matrix*, a bi-infinite matrix $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star\times\Sigma^\star}$ with rows and columns indexed by strings in $\Sigma^\star$. In particular, a well-known result states that $\mathbf{H}_f$ has rank at most n if and only if there exists a PSR $\mathcal{A}$ with n states satisfying $f_\mathcal{A} = f$ [Carlyle and Paz, 1971; Fliess, 1974]. The Hankel matrix $\mathbf{H}_f$ is tightly related to the *system dynamics matrix* (SDM) of the stochastic process described by f [Singh *et al.*, 2004], but while the entries of the Hankel matrix represent *joint* probabilities over prefixes and suffixes, the corresponding entry in the SDM is the *conditional* probability of observing a suffix given the prefix. An empirical estimate of the Hankel matrix can be obtained given a finite set of prefixes and suffixes. The singular value decomposition can then be used to recover a PSR (see [Boots *et al.*, 2011] for details).

2 Option Duration Models

We are interested in the dynamics of an agent interacting with an MDP M via a set of options Ω. Recall that in this setting the agent is not allowed to perform primitive actions, and options must be executed until termination. We are interested in considering situations where the *duration* of an option constitutes an informative statistic about the state of the MDP. Hence, the history of the agent's interaction with an MPD will be given by a trajectory consisting of option-duration pairs: $(\omega_1, d_1)(\omega_2, d_2)\cdots(\omega_t, d_t)$, with $\omega_i \in \Omega$, $d_i \in \mathbb{N} = \{1, 2, \ldots\}$. Focusing on the sequence of options and termination/continuation events, we have a discrete dynamical system with observations from $\Omega \times \{\sharp, \bot\}$, where $\sharp$ (*sharp*) denotes continuation and $\bot$ (*bottom*) denotes termination. The previous trajectory in this new dynamical system looks as follows:

$$(\omega_1, \sharp, \ldots, \omega_1, \sharp, \omega_1, \bot, \omega_2, \sharp, \ldots, \omega_2, \sharp, \omega_2, \bot, \ldots) = (\omega_1, \sharp)^{d_1-1}(\omega_1, \bot)(\omega_2, \sharp)^{d_2-1}(\omega_2, \bot)\ldots$$

Formally, we are mapping a dynamical process with trajectories in $(S \times A)^\star$ (representing the interaction of the agent with the MDP), to a process with trajectories in $(\Omega\times\{\sharp, \bot\})^\star$ representing the duration of option execution. This mapping induces a new dynamical system, whose properties might be useful for planning with options in the original system.

We now show that the probability distributions over the duration of options can be compactly represented in the form of a PSR. Let $s_0 \in S$, $\omega = \langle I, \pi, \beta\rangle$, and $d > 0$ be an integer. We write $\delta(s_0, \omega)$ for the random variable representing the duration until termination of option ω from state s_0. We are interested in the following quantity:

$$\mathbb{P}[\delta(s_0, \omega) = d] = \sum_{\bar{s}\in S^d}\sum_{\bar{a}\in A^d} \mathbb{P}[s_0, a_0, s_1, a_1, \cdots, a_{d-1}, a_{d-1}, s_d, \bot] \ , \tag{3}$$

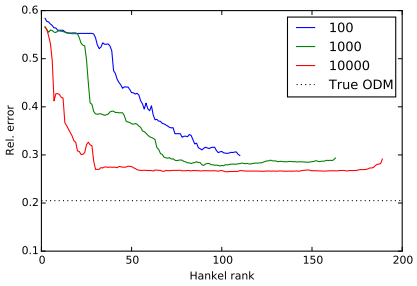
where $\bar{s} = s_1 \cdots s_d$ is the sequence of states traversed by ω, $\bar{a} = a_0 \cdots a_{d-1}$ is the sequence of actions performed by by ω, and $\bot$ denotes the option termination. With some algebra, it can be shown that summing this expression over $\bar{s}$ and $\bar{a}$ yields:

$$\mathbb{P}[\delta(s_0, \omega) = d] = \mathbf{e}_{s_0}^\top \mathbf{A}_{\omega,\sharp}^{d-1}\mathbf{A}_{\omega,\bot}\mathbf{1} \ , \tag{4}$$

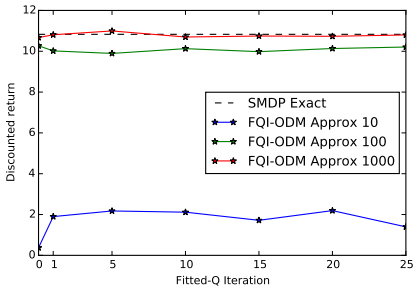
where we have used the following definitions: $\mathbf{e}_{s_0} \in \mathbb{R}^S$ is an indicator vector with $\mathbf{e}_{s_0}(s) = \mathbb{I}[s = s_0]$, $\mathbf{A}_{\omega,\sharp} \in \mathbb{R}^{S\times S}$ is a matrix with $\mathbf{A}_{\omega,\sharp}(s, s') = \sum_{a\in A}\pi(s, a)P(s, a, s')(1 - \beta(s'))$, $\mathbf{A}_{\omega,\bot} \in \mathbb{R}^{S\times S}$ is a matrix with $\mathbf{A}_{\omega,\bot}(s, s') = \sum_{a\in A}\pi(s, a)P(s, a, s')\beta(s')$, and $\mathbf{1} \in \mathbb{R}^S$ is a vector of ones. More generally, we can prove the following:

Theorem 1. *Let M be an MDP with n states, Ω a set of options, and $\Sigma = \Omega \times \{\sharp, \bot\}$. For every distribution α over the states of M, there exists a PSR $\mathcal{A} = \langle \boldsymbol{\alpha}, \mathbf{1}, \{\mathbf{A}_\sigma\}\rangle$ with at most n states that computes the distributions over durations of options executed from a state sampled according to α.*

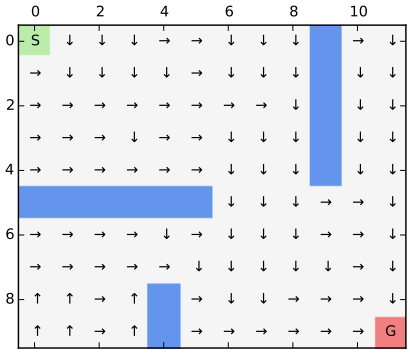
We will call any PSR computing distributions over durations of options an *option duration model* (ODM).



(a) Predictive accuracy: relative error versus rank for different sample sizes



(b) Planning with a learned ODM



(c) Room environment overlaid with an optimal policy over options

3 Experiments

We first assess the learnability of our model in practice using a gridworld environment. We use a 4-connected grid with four actions representing the cardinal directions (NEWS). Unless the current state is a "wall" each action moves the agent one step in the specified direction with probability 0.9, and remains in the current state with probability 0.1. We also define one option for each cardinal direction. These options take as many steps as possible in the specified direction until they hit a wall, at which point the option terminates. A uniform random exploration policy is used for sampling 10000 episodes in which five options are executed up to termination. We also collected a test set consisting of 10000 trajectories of eight options sequences. We evaluate the prediction accuracy by computing the relative error over the estimated remaining number of steps in the currently executing option. For each test trajectory, we picked a time index uniformly at random and conditioned the learned ODM on the history up to this point. These random split points were then kept fixed throughout all evaluations. Figure 1a shows that the prediction accuracy increases as the dimension of the ODM gets larger. More samples also allow for better predictions. Note that since the prediction task is inherently stochastic, even the true ODM cannot achieve zero relative error.

3.1 Planning

We use the Fitted-Q iteration (FQI) algorithm of Ernst *et al.* [2005] for planning over a learned ODM. We make state directly over the ODM state vector updated at each step with the corresponding operator (continuation or termination) according to the linear form in (2). A gridworld environment with obstacles is used for evaluation and once again, any of the four actions can fail with probability 0.1 in every state. An immediate reward of 100 is obtained at the goal and collisions with the walls are penalized by -10. Taking a primitive step does not incur an immediate cost but the length of the trajectories affect the cumulative reward through a discount factor of 0.9. A dataset of 1000 trajectories of eight options sequences was collected with a discrete uniform policy over options. For each curve shown in 1b, we use our dataset to learn an ODM and plan over it. We evaluate the performance of the greedy policy by taking 100 Monte-Carlo estimates in the simulated environment. Given the true underlying MDP and a set of options, we can compute the resulting Semi-Markov Decision Process (SMDP) (see p. 26 of Sutton *et al.* [1999]) and solve it using value iteration. The expected discounted cumulative return in the SMDP serves as our baseline. Figure 1b shows that an optimal policy can be obtained using 1000 trajectories and one step of FQI. Interestingly, it seems that even when using an imperfect model (such as the one built with 100 trajectories in fig. 1b), we can still recover a near-optimal policy.

References

B. Boots, S. Siddiqi, and G. Gordon. Closing the learning planning loop with predictive state representations. *International Journal of Robotic Research*, 2011.

M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. *International Conference on Machine Learning (ICML)*, 2005.

J. W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computer Systems Science*, 1971.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.*, 6:503–556, December 2005.

M. Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 1974.

J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing*, 1977.

M.L. Littman, R.S. Sutton, and S. Singh. Predictive representations of state. *Neural Information Processing Systems (NIPS)*, 2002.

A. Machado, M. T. Malheiro, and W. Erlhagen. Learning to time: A perspective. *Journal of the Experimental Analysis of Behavior*, 2009.

M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. *International Conference on Machine Learning (ICML)*, 2004.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. *International Conference on Machine Learning (ICML)*, 2004.

S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. *Uncertainty in Artificial Intelligence (UAI)*, 2004.

J. Stober, R. Miikkulainen, and B. Kuipers. Learning geometry from sensorimotor experience. *Joint Conference on Development and Learning and Epigenetic Robotics*, 2011.

R. S Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

B. Wolfe and S. Singh. Predictive state representations with options. *International Conference on Machine Learning (ICML)*, 2006.