RLDM 2013

Extended Abstracts

October 25 - October 27, 2013
Princeton University
Princton, NJ, USA
rldm.org

TABLE OF CONTENTS

Path Integral Stochastic Optimal Control for Reinforcement Learning

Farbod Farshidian
Institute of Robotics and Intelligent Systems
ETH Zürich
Zürich, 8092, Switzerland
farshidian@mavt.ethz.ch

Jonas Buchli
Institute of Robotics and Intelligent Systems
ETH Zürich
Zürich, 8092, Switzerland
buchlij@ethz.ch

Abstract

Path integral stochastic optimal control based learning methods are among the most efficient and scalable reinforcement learning algorithms. In this work, we present a variation of this idea in which the optimal control policy is approximated through linear regression. This connection allows the use of well-developed linear regression algorithms for learning of the optimal policy, e.g. learning the structural parameters as well as linear parameters. In path integral reinforcement learning, Policy Improvement with Path Integral (PI^2) algorithm is one of the most efficient and most similar algorithms to the algorithm we propose here. However, in contrast to the PI^2 algorithm that relies on the Dynamic Movement Primitive (DMPs) to become a model free learning algorithm, our proposed method is formulated for an arbitrary parameterized policy represented by a linear combination of nonlinear basis functions. Additionally, as the duration and the goal of the task is part of the optimization in some tasks like shortest-time path optimization problem, our proposed method can directly optimize these quantities instead of assuming them to be given, fixed parameters. Furthermore PI^2 needs a batch of rollouts for each parameter update iteration whereas our method can update after just one rollout. The simulation result in this work shows that a simple implementation of our proposed method can at least perform as well as PI^2 despite only using 'out-of-the-box' regression and a 'naive' sampling strategy. In this light, the here presented should only be considered as a preliminary step in the development of our new approach which addresses some issues in the derivation of previous algorithms. Basing the development on this improvements, we believe that this work will ultimately lead to more efficient learning algorithms.

Keywords: reinforcement learning, stochastic optimal control, path integral, linear regression approximation.

Acknowledgements

This research is supported by the Swiss National Science Foundation through a SNSF Professorship Award to Jonas Buchli and the SNSF National Centre of Competence in Research Robotics.

1 Introduction

Learning motor control is one the essential components for versatile autonomous robots. In order to guide the learning, we need to define a cost (or reward) function that mirrors the requirements for a given task. Therefore learning motor control can be considered as a constraint optimization problem where the constraints are in general a set of stochastic differential equations plus some limits on the joint variables and torques. This naturally leads to formulating learning motor control in a stochastic optimal control framework, where the main difference is the availability of a model (optimal control) vs. no model (learning). Taking a model based optimal control perspective and then developing a model free reinforcement learning algorithm based on an optimal control framework has proven very successful. Under the assumption of stochastic dynamics the key for the step from model based control to model free learning is a path integral formulation of optimal control. Recently, algorithms based on these developments [10, 4] have shown superior performance in practical applications [9, 1].

Using path integral to solve the optimal control problem was introduced by Kappen in [3]. He demonstrates how path integral allows to replace the backward computation, by forward computation of a diffusion process. But his formulation is not model-free and also it is restricted to a special class of systems i.e. system with state independent control transition. In Theodorou et al. [10], the generalized path integral formulation (the control transition can be a function of the states) is used to derive a model-free, iterative learning algorithm named Policy Improvement with Path Integral (PI^2). The original formulation of PI^2 is based on Dynamic Movement Primitives (DMPs) [2]. The DMPs add an artificial limitation to the optimization problem by restricting the space of the achievable trajectories. DMPs also introduce two open algorithmic parameters i.e. the goal and duration of movement. In [9] an approach to learn the goal parameter is presented, but still the duration of movement is an open parameter. In [7] a variation of PI^2 is presented in which the control input is directly parameterized. However this variation is not model-free (the control transition is needed) and suffers from over parameterization. To solve this problem as proposed in [1] a fast intermediate system between the real system input and parameterized input is needed (the same version of PI^2 is used here for the sake of comparison). In addition to the policy representation issue PI^2 needs a batch of rollouts for each iteration of parameter update which is not ideal for learning on real systems e.g. robots. Here every additional roll-out is time consuming and leads to unnecessary wear and tear. Also PI^2 relies on a task-dependent heuristic formula to sum up the time-indexed policy parameter vector to one time-independent parameter vector. In contrast to PI^2 our proposed algorithm addresses all these issues in a more systematic way.

In this work, we follow up on the idea of path integral stochastic optimal control as a basis for formulating a learning algorithm. However, instead of confining policy parametrization to the DMPs, we utilize an arbitrary linear parametrization for optimal policy by using linear combinations of nonlinear basis functions. We also try to avoid any kind of task dependent heuristic in order to develop a more general learning algorithm. As will be detailed, the estimation of the optimal control policy can be transformed into a linear regression problem. This connection to the linear regressions gives the opportunity of using well developed algorithms for learning both the model structure and the parameters. This approach also makes it possible to update parameter vector after each rollout (Although in current implementation this ability is not used).

2 Path Integral Stochastic Optimal Control Learning Algorithm

In this section, we are going to demonstrate how the problem of estimating the optimal policy boils down to a linear regression problem. For this reason, we will briefly introduce the path integral stochastic optimal control. Then, we will derive our main formula which is a linear regression optimal control policy.

Path Integral Stochastic Optimal Control – First, we briefley introduce the basics of path integral stochastic optimal control (for more details refer to [3, 4, 10]). We assume a finite horizon cost function

$$J(x_i, t_i) = \min_{u(t \to t_f)} C\left(x_i, t, u\left(t_i \to t_f\right)\right) \tag{1}$$

$$C\left(x_i, t_i, u\left(.\right)\right) = E_{\tau|x_i}\left\{\phi(x(t_f)) + \int_{t_i}^{t_f}\left(q(x,t) + \frac{1}{2}u^T R u\right) dt\right\}$$

where x is the state vector and u is the control input vector. The trajectory $\tau|x_i$ is generated by the stochastic dynamics modelled by

$$\dot{x} = f(x,t) + g(x)(u + \varepsilon) \qquad \varepsilon \sim N(0, \Sigma) \tag{2}$$

with initial condition $x(t_i) = x_i$, control trajectory $u(.)$, and zero mean Gaussian input noise ε with covariance Σ. The solution to this optimization problem can be derived by solving the stochastic Hamilton-Jacobi-Bellman (HJB) equation which is a nonlinear second order Partial Differential Equation (PDE). This PDE can be transformed into a linear second order PDE through the transformation $J = -\lambda \log \Psi$ and assumption $\Sigma = \lambda R^{-1}$. Using the Feynman-Kac formula, the

solution to this linear PDE can be found by simulating random paths of the stochastic process which is associated with the uncontrolled noise driven system. After several steps of calculation, the control input can also be derived by a path integral formulation. Equation (3) shows this relation to the path integral. In this equation g_c is the nonzero vertical partition of the matrix g and $Q(\tau|x)$ is the accumulated cost (without the control cost) on trajectory τ which starts form state x and evolves according to equation (4) (uncontrolled noise driven system).

$$\begin{aligned} u^*(x,t) =& T(x)\frac{E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\varepsilon_t\right\}}{E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\right\}} \\ T(x) =& R^{-1}g_c^T\left(g_cR^{-1}g_c^T\right)^{-1}g_c \\ Q(\tau|x) =& \left(\phi(x(t_f)) + \int_t^{t_f} q(x',t')dt'\right)_{\tau|x} \\ \dot{x'} =& f(x',t') + g(x')\varepsilon \end{aligned} \tag{3,4}$$

Equation (3) can also be written in the form of equation (5). This equation will be used in the next section.

$$E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\left(u^*(x,t) - T(x)\varepsilon_t\right)\right\} = 0 \tag{5}$$

Proposed Learning Algorithm – Using a parameterized policy makes the learning problem tractable in high dimension. Among different possible choices for function approximation a linear combination of basis functions, $\theta^T\Phi(x,t)$, represents theoretical and practical advantages. In general the basis functions $\Phi(x,t)$ of these models can be a nonlinear function of any set of features from time and states. In addition, any approximation problem needs a suitable metric to calculate the difference between the approximated and real optimal solution. Equation (6) shows the sum square error criterion, $L(\theta)$, which is calculated over an arbitrary distribution (ideally the optimal trajectory distribution) for approximating the optimal control input. Here $\| \ . \ \|_2$ is the norm two of the vector.

$$L(\theta) = E_x\left\{\| \ u^*(x,t) - \theta^T\Phi(x,t) \ \|_2^2\right\} \tag{6}$$

The gradient of $L(\theta)$ is equal to zero at the optimal solution which implies equation (7).

$$\frac{\partial L(\theta^*)}{\partial\theta} = E_x\left\{\left(u^*(x,t) - \theta^{*T}\Phi(x,t)\right)\Phi(x,t)^T\right\} = 0 \tag{7}$$

By multiplying and dividing $E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\right\}$ and further steps we omit for brevity we get

$$E_x\left\{\frac{1}{E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\right\}}E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\left(u^*(x,t) - \theta^{*T}\Phi(x,t)\right)\Phi(x,t)^T\right\}\right\} = 0 \tag{8}$$

by adding and subtracting the term $T(x)\varepsilon_t$ from the previous expression we derive the following expression

$$E_x\left\{\frac{1}{E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\right\}}E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\left(\underline{u^*(x,t) - T(x)\varepsilon_t} + T(x)\varepsilon_t - \theta^{*T}\Phi(x,t)\right)\Phi(x,t)^T\right\}\right\} = 0 \tag{9}$$

After several further steps and using equation (5) it can be shown that the underlined term will vanish. Therefore, we get

$$E_xE_{\tau|x}\left\{\frac{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)}{E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\right\}}\left(\theta^{*T}\Phi(x,t) - T(x)\varepsilon_t\right)\Phi(x,t)^T\right\} = 0 \tag{10}$$

which is equivalent to the following minimization problem

$$\theta^* = \arg\min_\theta E_xE_{\tau|x}\left\{\frac{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)}{E_{\tau|x}\left\{\exp\left(-\frac{1}{\lambda}Q(\tau|x)\right)\right\}} \ \| \ \theta^T\Phi(x,t) - T(x)\varepsilon_t \ \|_2^2\right\} \tag{11}$$

Equation (11) shows that the problem of finding the optimal parameterized policy is reduced to a Weighted Linear Regression (WLR) problem.

To turn this method into a model free algorithm, the matrix $T(x)$ should be omitted from regression. This can be achieved by assuming that the real system dynamics is augmented by a square MIMO (Multiple-Input Multiple-Output) linear system in which its states are the real system control input. This system should have a unique output-to-input gain and faster dynamics than the real, underlying system and does therefore not interfere with the real system dynamics. In this case the real system control inputs become part of the states of the augmented system and it causes matrix $T(x)$ to

Table 1 Path Integral Stochastic Optimal Control (PISOC) Learning Algorithm:

Given: Initial parameter θ, Exploration noise covariance, Linear model structure $u = \theta^T \Phi(x,t)$, Cost function.
while Convergence of the trajectory cost **do**
- Create K rollouts through the system in equation (2).
- Compute S in equation (12) for each rollout.
- Compute $minS$ and $maxS$ over K rollouts.
- Compute $\tilde{S}$ in equation (12) for each rollout.
- Update θ by solving the minimization in equation (12) over K rollouts.
- Anneal the exploration noise.

end while

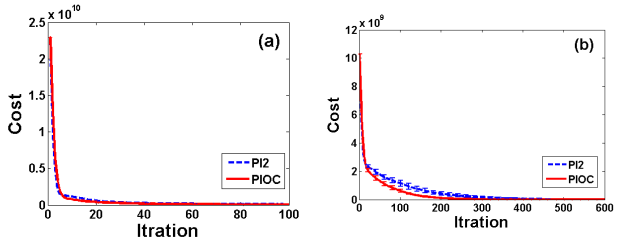


Figure 1 The evaluation results, the cost versus iteration for (a) mass point (b) 50-link arm

depend only on the input matrix of the linear system which is a invertible square matrix. This means $T(x) = I$. This simplification borrows the idea of the adiabatic elimination technique in physics [1].

Note that the sampling distribution in equation (11) comes from the noise driven system. This can cause two major problems. First, we expect that the performance of the learning agent increases during the time which obviously contradicts with sampling from noise driven system. Second, it causes the probability of finding good trajectories to decrease exponentially as the dimension of the problem increases. This makes the learning problem intractable in high dimension and real word problems. One possible remedy is using the importance sampling method. In this method, instead of sampling the whole solution space, samples are taken from a suitable sampling distribution. One natural choice for current learning problem will be sampling from active system where the control inputs are computed as the last estimation of the optimal control inputs with respect to the samples collected so far. The only remaining issue is to add importance sampling correction weight to the regression problem. By Gaussian noise assumption the final regression problem is as equation (12). Here the τ_a illustrates that the sampling system is the active system. In equation (12) $\tilde{S}$ is the normalized version of S. This accomplished by choosing the $\lambda = \frac{maxS - minS}{10}$ and multiplying the nominator and denominator by $exp(\frac{10 minS}{maxS - minS})$. Table 1 illustrates the pseudo code for a simple learning algorithm based on equation (12).

$$\theta^* = \min_{\theta} E_x E_{\tau_a|x} \left\{ \frac{\exp\left(-\tilde{S}(\tau_a|x)\right)}{E_{\tau_a|x}\left\{\exp\left(-\tilde{S}(\tau_a|x)\right)\right\}} \parallel \theta^T \Phi(x,t) - \varepsilon_t \parallel_2^2 \right\} \tag{12}$$

$$\tilde{S}(\tau_a|x) = 10 \times \frac{S(\tau_a|x) - \min_{\tau_a|x} S(\tau_a|x)}{\max_{\tau_a|x} S(\tau_a|x) - \min_{\tau_a|x} S(\tau_a|x)}$$

$$S(\tau_a|x) = \left(\phi(x(t_f)) + \int_t^{t_f} \left(q(x',t') + \frac{1}{2} u^T R u + u^T R \varepsilon\right) dt'\right)_{\tau_a|x}$$

3 Simulation

In this section the algorithm in the Table 1 is used for learning the two different tasks. Both of the evaluation are the via point and reaching task combination in the 2D environment. The first on is a single point mass and the second one is a 50-link planar arm. We compare our solution against a modified version of PI^2 algorithm for the performance comparison. The here used variant of PI^2 does not use the DMPs for policy parameterization. Whereas it directly parameterizes the control input (i.e. as our proposed algorithm). In both of the algorithms, the exploration noise, number of rollouts, and other open parameters are the same. The control input is represented by linear combination of time-based Gaussian kernels.

Point Mass – The first evaluation considers learning the optimal policy for controlling a point mass moving on a horizontal plane. The task is to start from a particular point, then to pass through another particular point in a specific time, and finally to reach to a goal point in the predefined time with zero velocity. We also try to minimize the control input quadratic cost. In both algorithms 15 trials per update are considered which consist of 10 new trials and the best 5 trials form previous updates. The covariance of the noise is set to 20 which is decreased linearly over parameter update iterations. The initial value for the parameter vector is set to zero. Figure 1.a shows the cost of task versus iteration for both algorithms. The figure shows that the both algorithms have comparable performances.

Robotic Arm – In the second evaluation, a 50-link planar robot is considered. The goal of the task is the same as in the previous example but this time the end-effector position and velocity is considered. The learning setup is also the same. Figure 1.b shows the result of learning curve. The results show that the proposed algorithm performs slightly better.

4 Discussion

As the simulation results demonstrate, the performance of current implementation of equation (12) is not significantly better than the PI^2. The results just confirm that a simple implementation of this linear regression based learning algorithm can work at least as well as PI^2. We consider this work as a basis for future development, as it addresses some issues of previous developments and it will allow for more efficient learning algorithms. There are a few interesting aspects of the proposed algorithm that we are discussing in the following.

– One of the advantages of the formulation in equation (12) is that it transforms the problem of the learning optimal control in to a linear regression problem which allows us to use the well developed framework and tools of regression.

– While we have not demonstrated this fact here, the structural parameters can be learned automatically. There exist a number of algorithms in the framework of linear model for regression which make it possible to automatically adapt the distribution and number of the base functions. Although it is possible to use the same methods for PI^2 algorithm, some practical considerations in the implementation of PI^2 will slow down structural parameter learning. Before starting to describe the reason, note that one of the difference between these two algorithm is the way they add noise to the system for exploration. In our proposed algorithm (like gradient based policy search) the noise is added directly to the action while in PI^2, exploration noise is added to the parameter vector (like stochastic gradient descend methods and evolutionary algorithms). As shown in [8] it is better not to add time dependent noise to the parameter vector but just to add a constant perturbation during each trial. This consideration will cause that for a single update of structural parameters, we require the information of several different rollouts. Whereas in the new formulation noise is varied during execution of a rollout which makes it possible to update the structural parameters after each rollout. In general we believe that adding noise directly to the control action increases the amount of information we can retrieve from the system. Although some may argue that it is not suitable for real systems to have non smooth input but it would be fine for simulation phase. In the case of real system experiment, we can assume a low pass filter before the system input which smooth out the control input signal. But again even by this assumption we can add a time varying noise in each rollouts (It would be colored noise not a white noise).

– According to the current formulation it is possible to do an update after each rollout. Consequently it can be expected that the accumulative reward during the agent life will increase quicker. Because not only the living policy is improved constantly but also the sampling distribution is modified.

– In the inference related methods for policy search, it is common to introduce the probability of reward in order to transform the optimization problem into inference problem [5, 6]. This reward probability is proportional to the exponential of the negative trajectory cost. Our proposed method lends support to this assumption. As shown, the problem of finding optimal parameter vector for arbitrary linear approximation of the optimal control input will be WLR, in which the weights are the reward probability. By the assumption of the Gaussian noise, The proposed WLR problem is equivalent to the maximum-likelihood estimation problem for a parameterized policy where the a-posterior distribution is proportional to the reward probability.

As future work, we will test this algorithm on real robotic systems. We also aim to implement a better algorithm for equation (12). This implementation should be able to learn the structural parameters (e.g. number of base functions and their distribution) as well as the linear parameters. It also should be able to update the parameterized policy after each rollout.

References

[1] J. Buchli, F. Stulp, E.A. Theodorou, and S. Schaal. Learning variable impedance control. *The International Journal of Robotics Research*, 2011.

[2] A.J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems*, 15:1523–1530, 2002.

[3] H.J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005(11), 2005.

[4] H.J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2), 2012.

[5] G. Neumann. Variational inference for policy search in changing situations.

[6] K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Proceedings, International Conference on Robotics Science and Systems (RSS 2012)*, 2012.

[7] E. Rombokas, E.A. Theodorou, M. Malhotra, E. Todorov, and Y. Matsuoka. Tendon-driven control of biomechanical and robotic systems: A path integral reinforcement learning approach. In *ICRA*, 2012.

[8] F. Stulp and O. Sigaud. Policy improvement methods: Between black-box optimization and episodic reinforcement learning. hal-00738463, 2012.

[9] F. Stulp, E.A. Theodorou, and S. Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *Robotics, IEEE Transactions on*, 28(6):1360–1370, 2012.

[10] E.A. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 2010.

The Advantage of Planning with Options

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

Temporally extended actions or options have primarily been applied to speed up reinforcement learning by directing exploration to critical regions of the state space. We show that options may play a critical role in planning as well. To demonstrate this, we analyze the convergence rate of Fitted Value Iteration with options. Our analysis reveals that for pessimistic value function estimates, options can improve the convergence rate compared to Fitted Value Iteration with only primitive actions. Furthermore, options can improve convergence even when they are suboptimal. Our experimental results in two different domains demonstrate the key properties from the analysis. While previous research has primarily considered options as a tool for exploration, our theoretical and experimental results demonstrate that options can play an important role in planning.

Keywords: Options, Temporally Extended Actions, Macro-Actions, Approximate Dynamic Programming, Fitted Value Iteration

Acknowledgements

1 Introduction

Although temporally extended actions or options have been primarily studied as a method for directing exploration, options may provide a complementary tool for efficient planning [5, 4]. Under most analyses of iterative planning algorithms, one iteration corresponds to planning one additional timestep into the future. On the other hand, by performing a single iteration with temporally extended actions, one iteration could instead correspond to planning several timesteps into the future. We derive bounds that help us reason about when approximate dynamic programming with temporally extended actions may converge faster than approximate dynamic programming with only primitive actions.

The options framework is appealing for investigating planning with temporally extended actions. For one thing, the class of options includes both primitive actions as well as a wide range of temporally extended actions, and many of the well-known properties of Markov Decision Processes generalize when arbitrary options are added (e.g., Value Iteration and Policy Iteration still converge [5]). In addition, much effort has gone into algorithms that learn "good" options for exploration. These algorithms may produce options that are also useful for planning. Lastly, options allow for greater flexibility when modeling problems where actions do not have the same temporal resolution. For example, in inventory management problems where placing orders may not occur at regular intervals. Thus, options are an important candidate for investigating planning with temporally extended actions.

We argue that planning can benefit from using options. To show this, we introduced a generalization of the Fitted Value Iteration (FVI) algorithm that incorporates samples generated by options. When the given set of options contains the primitive actions, our generalized algorithm converges at least as fast as FVI with only primitive actions. Then we develop precise conditions where our generalized FVI algorithm converges faster with options than with only primitive actions. These conditions turn out to depend critically on whether the iterates produced by FVI underestimate the optimal value function. Finally, our experimental results in two different domains demonstrate that the convergence rate of planning with options can be significantly faster than planning with only primitive actions. However, as predicted by our theoretical analysis, the improvement in convergence only occurs when the iterates of our algorithm underestimate the optimal value function, which can be controlled in practice by setting the initial estimate of the optimal value function pessimistically. Our analysis of FVI suggests that options can play an important role in planning by inducing fast convergence.

2 Background

A Markov Decision Process (MDP) is defined by a is a set of states X, a finite set of primitive actions A, a mapping P from state-action pairs to probability distributions over states, a mapping R from state-action pairs to bounded reward distributions, and a discount factor $\gamma \in [0, 1)$. Let $B(X; V_{\text{MAX}})$ denote the set of functions with domain X and range bounded by $[-V_{\text{MAX}}, V_{\text{MAX}}]$ and $M(X)$ the set of all probability measures on X. The objective of planning in an MDP is to derive a policy π that maximizes $V^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t(x_t, \pi(x_t)) | x_0 = x, \pi\right]$, where x is the long-term value of following π starting in state x. We denote an optimal policy by π^* and $V^* = V^{\pi^*} = \max_\pi V^\pi$. The MDP Bellman operator $\mathcal{T}$ performs the backup $\mathcal{T}V(x) = \max_{a \in A} R(x, a) + \gamma \int P(y|x, a)V(y)dy$, which defines the popular value iteration algorithm.

The Primitive action Fitted Value Iteration (PFVI) algorithm is a generalization of value iteration to handle MDPs with large or continuous state spaces. PFVI runs iteratively producing a sequence of $K \geq 1$ estimates $\{V_k\}_{k=1}^K$ of the optimal value function and returns a policy π_K that is greedy with respect to the final estimate V_K. During each iteration k, the algorithm computes a set of empirical estimates $\hat{V}_k$ of $\mathcal{T}V_{k-1}$ for N states, and then fits a function approximator to $\hat{V}_k$. To generate $\hat{V}_k$, N states $\{x_i\}_{i=1}^N$ are sampled from a distribution $\mu \in M(X)$. For each sampled state x_i and each primitive action $a \in A$, L next states $\{y_{i,j}^a\}_{j=1}^L$ and rewards $\{r_{i,j}^a\}_{j=1}^L$ are sampled from the MDP simulator $\mathbb{S}$. For the k^{th} iteration, the estimates of the Bellman backups are computed by $\hat{V}_k(x_i) = \max_{a \in A} \frac{1}{L} \sum_{j=1}^L \left(r_{i,j}^a + \gamma V_{k-1}(y_{i,j}^a)\right)$, where V_0 is the initial estimate of the optimal value function given as an argument to PFVI. The k^{th} estimate of the optimal value function is obtained by applying a supervised learning algorithm $\mathcal{A}$, that produces $V_k = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^N \left| f(x_i) - \hat{V}_k(x_i) \right|^p$, where $p \geq 1$ and $\mathcal{F} \subset B(X; V_{\text{MAX}})$ is the hypothesis space of the supervised learning algorithm.

Munos and Szepesvári [3] showed that given an MDP, if we select probability distributions $\mu, \rho \in M(X)$, a positive integer p, a supervised learning algorithm $\mathcal{A}$ over a bounded function space $\mathcal{F}$ that returns the function $f \in \mathcal{F}$ that minimizes the empirical p-norm error, $V_0 \in \mathcal{F}$ an initial estimate of the optimal value function, and $\varepsilon > 0$ and $\delta \in (0, 1]$, then for any $K \geq 1$, with probability at least $1 - \delta$, there exists parameters N and L such that the policy π_K returned by PFVI satisfies

$$||V^* - V^{\pi_K}||_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} d_{p,\mu}(\mathcal{T}\mathcal{F}, \mathcal{F}) + \varepsilon + \gamma^{\frac{K+1}{p}} \left(\frac{2 \left\|V^* - V_0\right\|_\infty}{(1-\gamma)^2}\right) , \tag{1}$$

where $d_{p,\mu}(\mathcal{TF},\mathcal{F}) = \sup_{f\in\mathcal{F}} \inf_{g\in\mathcal{F}} \|\mathcal{T}f - g\|_{p,\mu}$ is the inherent Bellman error of $\mathcal{F}$ with respect to Bellman operator $\mathcal{T}$ and $C_{\rho,\mu}^{1/p}$ describes the smoothness of the MDP's future state distributions. The inherent Bellman error is a measure of how well the chosen hypothesis space $\mathcal{F}$ can represent $\hat{V}_k$ at each iteration. The first term in (1) is called the approximation error and corresponds to the error introduced by the inability of the supervised learning algorithm to exactly capture $\hat{V}_k$ at each iteration, while the second term, the estimation error, is due to using a finite number of samples to estimate $\hat{V}_k$. The last term is controlled by the number of iterations K of the algorithm. By increasing K the last term shrinks exponentially fast. This last term characterizes the convergence rate of the algorithm. The size of the discount factor γ controls the rate of convergence. Convergence is faster when γ is smaller. Unfortunately, γ is part of the problem definition. However, because options execute for multiple timesteps, an option can have an effective discount factor that is smaller than γ.

An MDP paired with a set of options $\mathcal{O}$ defines a Semi-Markov Decision Process (SMDP). An option o is defined by a set of states $\mathcal{I}_o$ that o can be initialized from, a policy π_o defined over primitive actions followed during the lifetime of o, and a mapping $\beta_o : X \to [0,1]$ that determines the probability that o will terminate in a given state [5]. Options have primarily been applied to direct exploration in reinforcement learning. However, Sutton et al. [5] and Silver and Ciosek [4] provide experimental results demonstrating that options can speed up planning in finite state MDPs, but these works did not apply options to tasks with continuous state spaces and there is no theoretical analysis of the convergence rate compared to planning with only primitive actions. We denote the transition probability matrix of an option o by P_o and the discounted transition probability matrix by $\widetilde{P}^o$. The SMDP Bellman operator $\mathbb{T}$ is defined by $\mathbb{T}V(x) = \max_{o\in\mathcal{O}} R(x,o) + \int \widetilde{P}^o(y|x)V(y)dy$.

We introduce a generalization of the FVI algorithm for the case where samples are generated by options (with primitive actions as a special case). The algorithm, Options Fitted Value Iteration (OFVI), takes the same arguments as PFVI, but operates on an SMDP simulator instead of an MDP simulator $\mathbb{S}$. When an option is sampled, $\mathbb{S}$ returns a termination state, cumulative reward, and duration that the option executed $\langle y_{i,j}^o, r_{i,j}^o, \tau_{i,j}^o\rangle \sim \mathbb{S}(x_i, o)$. Then the update resulting from applying the Bellman operator to the previous iterate V_{k-1} is estimated by $\hat{V}_k(x_i) \leftarrow \max_{o\in\mathcal{O}_{x_i}} \frac{1}{L}\sum_{j=1}^{L}\left[r_{i,j}^o + \gamma^{\tau_{i,j}^o} V_{k-1}(y_{i,j}^o)\right]$, and we apply a supervised learning algorithm to obtain the best fit in $\mathcal{F}$. The given simulator $\mathbb{S}$ differs from the simulator for PFVI. In addition to returning a next state and reward, $\mathbb{S}$ also returns the number of timesteps that the option executed before terminating. Otherwise the differences between PFVI and OFVI are minor and it is natural to ask if OFVI has similar finite-sample and convergence behavior compared with PFVI.

3 Theoretical Analysis

It turns out that, with no special assumptions about the given options or iterates of OFVI, it is possible to derive a bound that is similar to (1). We omit this result for brevity. In this section, we investigate when the convergence rate of OFVI is strictly faster than PFVI.

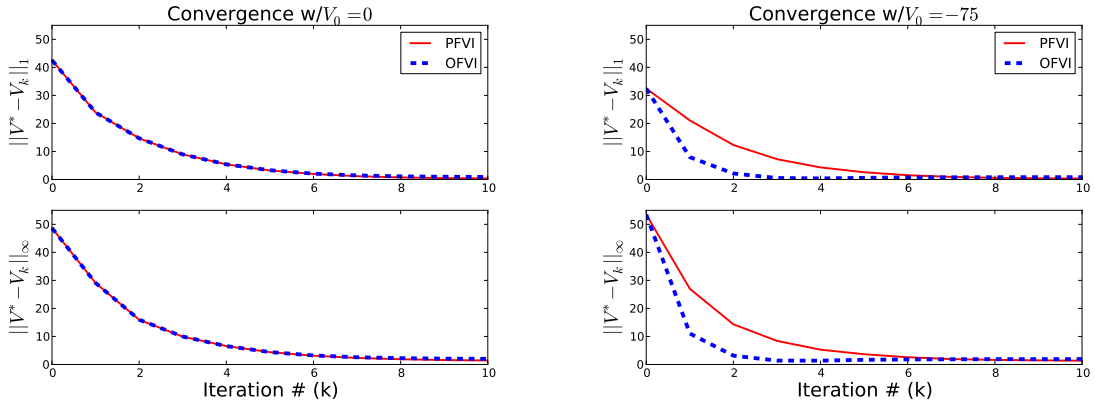
Faster convergence depends critically on the optimism or pessimism of the sequence of iterates produced by OFVI. We say that an estimate $V \in B(X; V_{\mathrm{MAX}})$ of the optimal value function is optimistic if $V(x) > V^*(x)$ for all $x \in X$, and we say that V is pessimistic if $V(x) \leq V^*(x)$ for all $x \in X$. It turns out that the SMDP Bellman operator $\mathbb{T}$ has the same convergence rate as the MDP Bellman $\mathcal{T}$ operator (although the SMDP Bellman operator can converge slightly slower in practice) when acting on entries of $V \in B(X; V_{\mathrm{MAX}})$ that are optimistic. This means that we can only expect OFVI to converge more quickly than PFVI when some of the iterates $\{V_k\}_{k=0}^K$ are pessimistic in at least part of the state space. For standard value iteration this is not a problem because we can set the initial estimate V_0 to be pessimistic and the fact that $\mathbb{T}$ is monotonic and converges to V^* ensures that every iterate is also pessimistic. The situation for OFVI is more complicated because of the algorithm's fitting step. However, Theorem 1 (below) describes when we can expect OFVI to converge faster than PFVI provided that the first few iterates happen to be pessimistic with respect to V^*.

Another important factor is the given set of options $\mathcal{O}$. We have already assumed that $\mathcal{O}$ contains the primitive actions in A. We need additional options that are somehow useful for planning. In the following theorem, we will assume that there exists a near-optimal option policy $\phi : X \to \mathcal{O}$ and that ϕ is temporally extended so that its effective discount factor $\bar{\gamma} < \gamma$.

Theorem 1. *For any $\varepsilon, \delta > 0$, $\bar{\gamma} \in (0,\gamma)$, $K \geq 1$, and $1 \leq Z \leq K$. Fix $p \geq 1$. Let $\rho, \mu \in M(X)$. Suppose there exists an option policy $\varphi : X \to \mathcal{O}$ that is α-optimal (i.e., $V^\varphi(x) > V^*(x) - \alpha$ for all $x \in X$) and $\widetilde{P}^\varphi \leq \bar{\gamma}P^\varphi$. Given $V_0 \in B(X, V_{\mathrm{MAX}})$, if the first Z iterates $\{V_k\}_{k=0}^Z$ produced by the algorithm are pessimistic (i.e., $V_k(x) \leq V^*(x)$ for all $x \in X$), then there exists positive integers N, L such that when OFVI is executed,*

$$||V^* - V^{\pi_K}||_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p}\left(d_{p,\mu}(\mathbb{T}\mathcal{F},\mathcal{F}) + 2\alpha\right) + \varepsilon + \left(\gamma^{K-Z+1}\bar{\gamma}^Z\right)^{1/p}\left(\frac{2\,\|V^* - V_0\|_\infty}{(1-\gamma)}\right) \tag{2}$$

holds with probability at least $1-\delta$.



(a) Optimistic ($V_0 \geq V^*$) (b) Pessimistic ($V_0 \leq V^*$)

Figure 1: Optimal Replacement Task: Convergence rates of PFVI and OFVI. (a) When the initial value function estimate is optimistic, there is no difference between the convergence rates of PFVI and OFVI. (b) However, when the value function estimate is pessimistic, OFVI converges faster than PFVI.

We omit the proof of Theorem 1 for brevity. In this theorem, we have assumed that there exists an option policy φ that is both near-optimal and temporally extended throughout the state space. The OFVI algorithm runs exactly as PFVI using the same parameters and requires no special prior knowledge or preparation besides setting V_0 to be pessimistic. For the results of the theorem to hold, at least the first Z iterates produced by the algorithm must be pessimistic. This condition is difficult to control in general, but it may be possible to control for specific applications and function approximation architectures.

As with (1), the bound in Theorem 1 has three terms: (1) approximation error, (2) estimation error, and (3) convergence error. The main advantage of Theorem 1 can be seen in the third term, which characterizes the convergence rate of the algorithm. The first Z iterations converge at a rate of $\bar{\gamma}$ instead of γ. Since $\bar{\gamma} < \gamma$, we can see that $\gamma^{K-Z+1}\bar{\gamma}^Z < \gamma^{K+1}$. In other words, when the conditions of Theorem 1 are met, OFVI converges faster than PFVI.

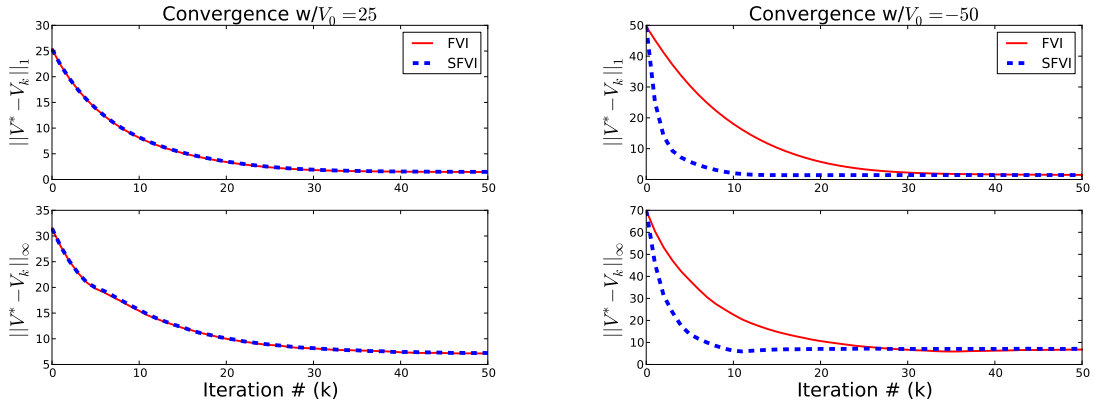
The main limitation is due to the first term, which controls the approximation error. This term is slightly worse than the approximation error in (1) due to our exploitation of the α-optimal option policy φ. However, this does not imply that the algorithm converges to a worse solution than (1). It reflects the fact that when such an option policy φ exists, convergence will be rapid up to a point. Once the algorithm has achieved a value function estimate that is close to φ, the convergence rate may slow because the SMDP Bellman operator may not be able to improve on the current estimate further by selecting temporally extended actions.

4 Experiments & Results

We compared PFVI and OFVI in two different tasks: (1) the optimal replacement problem considered in Munos and Szepesvári [3], and the Taxi domain introduced in Dietterich [1]. In both experiments, we see that options can dramatically improve convergence rates of FVI, but only when V_0 is pessimistic with respect to V^*.

In the optimal replacement problem, the agent selects from one of two actions K and R, whether to maintain a product (action K) at a maintenance cost $c(x)$ that depends on the product's condition x or replace (action R) the product with a new one for a fixed cost C. We used parameter values identical to those used by Munos and Szepesvári [3]. Similar to Munos and Szepesvári [3], we used polynomials to approximate the value function. Results presented here used fourth degree polynomials. The optimal policy keeps the product up to a point $\bar{x}$ and replaces the product once the state equals or exceeds $\bar{x}$. For OFVI, we introduced a single option that keeps the product up to a point $\bar{x}$ and terminates once the state equals or exceeds $\bar{x}$. For an optimistic initial value function, the behavior of PFVI and OFVI was almost identical (Figure 1a). With a pessimistic initial value function estimate, convergence of OFVI is significantly faster than PFVI (Figure 1b).

The Taxi task is to traverse a taxi to a passenger waiting in one of four landmarks, and then drop off the passenger at his desired location [1]. To approximate the value function, we used fourth order polynomial functions, but the state space was partitioned by landmarks and passenger locations so that the value function was approximated by a separate polynomial function for each partition. For OFVI, we introduced four options. Each option causes the agent to navigate to one of the four landmarks along the shortest path. Each option terminates either when it has reached its respective landmark or the option has exceeded the maximum trajectory length. Figure 2 shows average convergence rates for OFVI and PFVI with optimistic (Figure 2a) and pessimistic (Figure 2b) initial value functions. Similar to what was observed for



(a) Optimistic ($V_0 \geq V^*$) (b) Pessimistic ($V_0 \leq V^*$)

Figure 2: Taxi Task: Convergence rates of PFVI and OFVI. (a) When the initial value function estimate is optimistic, there is no difference between the convergence rates of PFVI and OFVI. (b) However, when the value function estimate is pessimistic, OFVI converges faster than PFVI.

the Optimal Replacement Task, in the Taxi Task, OFVI and PFVI have similar convergence rates when the initial value function is optimistic, but OFVI has a much faster convergence rate when the initial value function is pessimistic.

5 Conclusion

We have analyzed Fitted Value Iteration applied to SMDPs. Our analysis shows that when the value function estimate is pessimistic with respect to the optimal value function, the convergence rate of OFVI can be significantly faster than PFVI because OFVI can take advantage of temporally extended actions that have a smaller effective discount factor. Furthermore, options can improve convergence even when they are suboptimal. For other conditions, OFVI has a comparable convergence rate to PFVI.

Options may have other benefits for planning besides improving the convergence rate. For example, options may enable a planning algorithm to "skip over" regions of the state space with highly complex dynamics without impacting the quality of the planned policy. In partially observable environments, options may be exploited to decrease uncertainty about the hidden state by "skipping over" regions of the state space where there is large observation variance, or "testing" hypotheses about the hidden state. Options may also play an important role in robust optimization, where the dynamics of temporally extended actions are known with greater certainty than the dynamics of primitive actions.

References

[1] Thomas G Dietterich. The MAXQ method for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 118–126, 1998.

[2] Milos Hauskrecht. Planning with macro-actions: Effect of initial value function estimate on convergence rate of value iteration. Technical report, Brown University, 1998.

[3] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.

[4] David Silver and Kamil Ciosek. Compositional Planning Using Optimal Option Models. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, 2012.

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

Safe Reinforcement Learning Through Probabilistic Policy Reuse

Javier García
Departamento de Informática
Universidad Carlos III de Madrid
Avenida de la Universidad 30, 28911 Leganés. Spain
fjaviergp@gmail.com

Daniel Acera
Departamento de Informática
Universidad Carlos III de Madrid
Avenida de la Universidad 30, 28911 Leganés. Spain
100072822@alumnos.uc3m.es

Fernando Fernández
Departamento de Informática
Universidad Carlos III de Madrid
Avenida de la Universidad 30, 28911 Leganés. Spain
ffernand@inf.uc3m.es

Abstract

This work introduces *Policy Reuse for Safe Reinforcement Learning* (PR-SRL), an algorithm that combines Probabilistic Policy Reuse and teacher advices for safe exploration in dangerous and continuous state and action reinforcement learning tasks. The algorithm uses a progressive risk function which permits to identify the probability to end up in a fail from a given state. Such risk function is defined in terms of how far such state is from the state space known by the learning agent. Probabilistic Policy Reuse is used to safely balance the exploitation of actual learned knowledge, the exploration of new actions and the request of teacher advice in considered dangerous parts of the state space. Specifically, the π-reuse exploration strategy is used. Using experiments in the helicopter hover task, we show that the π-reuse exploration strategy reduces drastically the number of times that the learning system damages in training when compared with previous approaches, obtaining similar performance (in terms of the classical long-term accumulated reward) of the learned policy. We also show interesting results in to improve a basic walking behavior of the humanoid robot NAO.

Keywords: Safe Reinforcement Learning, Probabilistic Policy Reuse, Learning from Demonstration, Case Based Learning

Acknowledgements

This work has been partially supported by Spanish Government under project TIN2012-38079-C03-02, and by a subcontract from Universidad de Málaga under the project ITC-20111030.

1 Introduction

While most Reinforcement Learning (RL) tasks [6] are focused on maximizing a long-term cumulative reward, RL researchers are paying increasing attention also to the safety of the approaches (e.g., avoiding collisions, crashes, etc.) during the learning process [3, 2]. Thus, when using RL techniques in dangerous control tasks, an important question arises; namely, how can we ensure that the exploration of the state-action space will not cause damage or injury while, at the same time, learning (near-)optimal policies? The matter, in other words, is one of ensuring that the agent is able to explore a dangerous environment both safely and efficiently.

***P**olicy **I**mprovement through **S**afe **R**einforcement **L**earning* (PI-SRL) [2] is an algorithm for safe exploration in dangerous and continuous control tasks. Such a method requires a predefined (and safe) baseline policy, provided by a teacher, which is assumed to be suboptimal (otherwise, learning would be pointless). The method has shown the best performance in all the domains evaluated when compared with previous approaches, like the evolutionary RL approach selected winner of the helicopter domain in the 2009 RL Competition [4] and Geibel and Wysotzki's risk-sensitive RL approach [3]. PI-SRL is based in a risk function, $\varrho^B(s)$, that measures the risk of a state in terms of its similarity to previously visited (and secure) states in a case base, B. When the distance to the closer state in the case base is larger than a parameter θ, the risk is maximum, while the risk is minimal if such distance is lower than θ [2]. Therefore, in that work, the risk function is defined as a step function. However, to define the risk function in such a way has demonstrated that still may produce damages in the learning agent. The reason is that to follow the teacher advice only when the distance to the closest known state is larger than θ may be very late. Opposite, one would expect that the risk function is progressive. In such a way, while the limit of θ is approaching, the risk should start to grow, and the learning agent could start to use the teacher advice.

In this work we propose the use a progressive risk function that determines the probability to follow the teacher advice. To integrate such advice, Probabilistic Policy Reuse is used, specifically, the π-reuse explotation strategy. In its initial definition, the π-reuse strategy is an exploration strategy able to bias a new learning process with a previously acquired policy [1], but recent works have suggested also its use to incorporate teacher advice [8] and even human demonstrations [7]. We use this exploration strategy to incorporate the teacher advice in a safe exploration process. In this way, we minimize the number of times that the learning system damages in training, while maintaining the performance of the final policy achieved.

2 Safe Reinforcement Learning

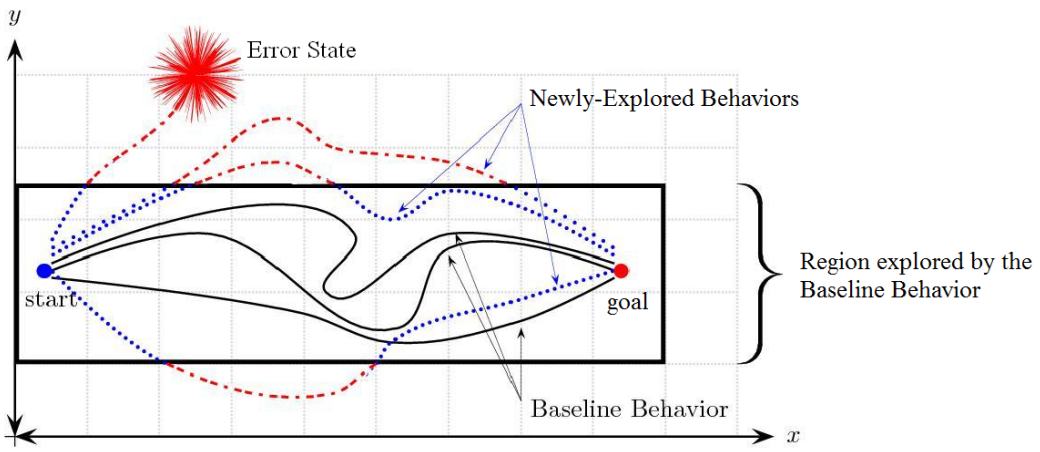


Figure 1: Exploration strategy based on the addition of small amounts of noise to baseline policy behavior.

The notion of *risk* considered in this paper is related to the fact that, during the exploration process, the agent can visit regions for which it has no information about how to act. As a result, the probability of incurring damage or injury is greatly increased. To illustrate this concept of risk, a navigation problem is presented in Figure 1. In this problem, small amounts of gaussian noise are added to a baseline policy to discover new trajectories to complete the task. However, the exploration of new trajectories leads the robot to unknown regions of the state space (the dashed red lines). The robot is assumed to be able to detect such situations with a risk function and, then, the baseline behavior is used to provide information about how to act.

In our approach, we represent a policy following a Case Based Reasoning Approach. A case-base is a set of cases $B = \{c_1 \ldots, c_\eta\}$ with $c_i =< s_i, a_i, V(s_i) >$, where the first element represents the case's problem part and corresponds to the

state s_i, the following element a_i depicts the case solution (i.e., the action expected when the agent is in the state s_i) and the final element $V(s_i)$ is the value function associated with the state s_i. Hence, the cases in B describe a **Case Based Policy** of the agent, π_B^θ, and its associated value function $V^{\pi_B^\theta}$. When the agent receives a new state s_q, the agent first retrieves the nearest neighbor to the s_q point in B according to some similarity metric and then the associated action is performed. In this paper we consider the Euclidian distance as similarity metric.

A *density threshold*, θ, is used to determine when a new case should be added to the memory. When the agent receives a new state s, it performs the action a_i of the case c_i for which $d(s, s_i) = \min_{1 \leq j \leq \eta} d(s, s_j) < \theta$ (known state). However, if the agent receives a state s where, by definition, the distance to any state in B is larger than θ (unknown state), no case is retrieved. Consequently, the action to be performed from that state is unknown to the agent. In that case, a baseline behavior, π_T, is used to support the exploration process conducted to obtain a better policy. In this step, π_T acts as a backup policy in the case of an unknown state with the intention of guiding the learning away from catastrophic errors or, at least, reducing their frequency. It is important to note that the baseline behavior cannot demonstrate the correct action for every possible state. However, while the baseline behavior might not be able to indicate the best action in all cases, the action it supplies should, at the very least, be safer than that obtained through random exploration.

2.1 Supporting the Exploration Process: the Risk Function

A main contribution of this work is the definition of a continuous risk function (Figure 2 (b)). Given a case base $B = \{c_1 \ldots, c_\eta\}$ composed of cases $c_i = (s_i, a_i, V(s_i))$, the risk for each state s is defined as Equation 1.

$$\varrho^B(s) = 1 - \frac{1}{1 + e^{\frac{k}{\theta}((\min_{1 \leq j \leq \eta} d(s,s_j) - \frac{\theta}{k}) - \theta)}} \tag{1}$$

Equation 1 allows us to obtain a smoother transition between risk-free states (i.e., known states) and risk states (i.e., unknown states). The parameter k has a double effect. On one hand, depending on its value, the width of the sigmoid function varies. A lower value of k implies a wider sigmoidal function. This results in a less aggressive exploration of the state space during the learning process since the baseline behavior advices are more frequently required (being able to affect negatively the final performance of the algorithm). On the other hand, the parameter k is used to displace the sigmoid function to the left, reducing in this way the probability of consider unknown states as known states. However, it is important to note that this probability does not disappear completely, i.e., this displacement allows also to keep the smooth transition to the right of the θ parameter. From Equation 1, the application of Probabilistic Policy Reuse to measure the advice of a teacher in safe RL is easy, by using the risk function $\varrho^B(s)$ as a transfer function. Therefore, the transfer rate depends on the safety of the learning agents: if safety is high, probability to use the advice of the teacher is very low, while if safety is low, such probability increases. This integration is explained next.

2.2 The PR-SRL Algorithm

Table 2 shows Safe π-reuse, a version of the π-reuse algorithm to incorporate the teacher advice in the exploration process. The main elements over the original π-reuse strategy are:

- the past policy Π_{past} is replaced by the baseline behavior π_T
- the new policy to be learned Π_{new} is replaced by the case base policy π_B^θ
- the parameter ψ is replaced by $\varrho^B(s)$
- no ϵ-greedy strategy is used, because actions are continuous. A random gaussian noise with standard deviation σ is used instead to generate exploratory actions

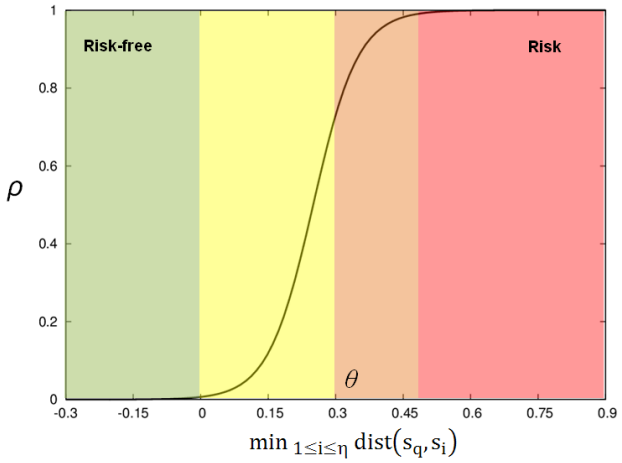
The algorithm builds a case for each step of an episode. For each new state s_h, the closest case $< s, a, V(s) > \in B$ is computed using the Euclidean distance metric (see line 05 in algorithm of Figure 2). At this point, the π-reuse strategy is followed, using the $\varrho^B(s)$ function as a transfer probability: with a probability of $\varrho^B(s)$ the policy of the baseline behavior π_T is followed, while with a probability of $1-\varrho^B(s)$, the action suggested by current base case policy is executed. Therefore, in areas far from the known states, the probability to use the baseline behavior advice is very high, while this advice is rarely used in known areas (see lines 07 and 08).

If the algorithm follows the baseline behavior, the action a_h performed is suggested by the baseline behavior, π_T, which defines safe behavior, and a new case $< s_h, a_h, 0 >$ is built (line 07). In this case, the state s_h is considered an unknown state. If the algorithm exploits the new policy, the action a_h is performed (line 08). In this case, the new case $< s, a_h, V(s) >$ is built replacing the action a corresponding to the closest case in $< s, a, V(s) > \in B$, with the new action a_h resulting from the application of random Gaussian noise to a. Thus, the algorithm only produces smooth changes in the cases of B where $a_h \sim a$. In this case, the state s_h is considered a known state. Finally, the reward obtained in the

Safe π-reuse (π_T, H, B, σ)

00 *listCasesEpisode* $\leftarrow \emptyset$.
01 $totalRwEpisode = 0$.
02 Set $h = 1$.
03 Set the initial state, s_h.
04 for h to H
05 Compute the case $< s, a, V(s) > \in B$ closest to the current state s_h
06 Set $\varrho^B(s_h) = 1 - \frac{1}{1+e^{\frac{k}{\theta}((\min_{1\leq j\leq \eta} d(s_h,s_j)-\frac{\theta}{k})-\theta)}}$
07 With a probability of $\varrho^B(s_h)$: $a_h = \pi_T(s_h)$, $c^{new} := (s_h, a_h, 0)$
08 With a probability of $1 - \varrho^B(s_h)$:
08 $a_h = rnd\_gaussian(\pi_B(s_h), \sigma)$, $c^{new} := (s, a_h, V(s))$
09 Execute a_h and receive the next state s'_h, and reward, $r(s_h, a_h)$
10 $totalRwEpisode := totalRwEpisode + r(s_h, a_h)$
11 $listCasesEpisode := listCasesEpisode \cup c^{new}$
12 Set $s_h \leftarrow s'_h$
13 Return $listCasesEpisode, totalRwEpisode$

(a)



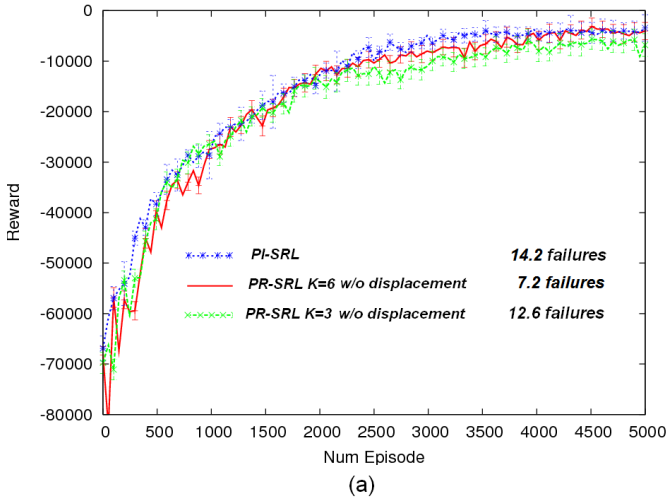
(b)

Figure 2: (a) Safe π-reuse Exploration Strategy for Safe Exploration of the State and Action Spaces. (b) Continuous risk function.

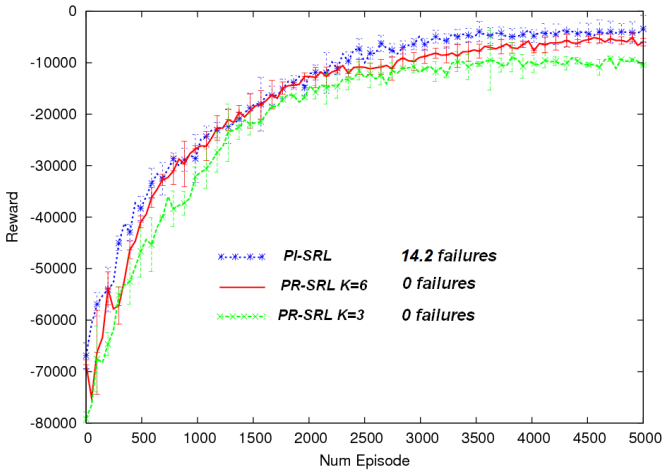
episode is accumulated, where $r(s_h, a_h)$ is the immediate reward obtained when action a_h is performed in state s_h (line 10) and the new case is added to the list of cases (line 11). PR-SRL uses the Safe π-reuse algorithm following iteratively the same steps of the *Improving the Learned Baseline Behavior Step* of PI-SRL described in [2], where the *Case Generation* step is replaced by the algorithm in Figure 2 (a).

3 Experimental Results

Firstly, we show the results in the classical generalized helicopter hovering domain [5] used in previous RL competitions. We propose to learn a near-optimal policy minimizing the failures, i.e., the helicopter crashes during the learning phase. The results of PR-SRL are compared with the results of PI-SRL, which in a previous work [2], where more details about the domain and previous experiments can be found, proved to have less failures (i.e., collisions, bankrupts) than an Evolutionary approach and a Risk-Sensitive approach. The experiments reported in Figure 3 (a) demonstrate that PR-SRL using a non-displaced continuous risk function with $k = 3$ and $k = 6$ generate failures. Additionally, the number of failures with PR-SRL $k = 3$ is higher since there is a higher probability to consider unknown states as known states. However, in both cases, PR-SRL obtains a lower number of failures than PI-SRL, proving that using the continuous risk function is better than using the discrete one. Figure 3 (b) shows the performance of PI-SRL and PR-SRL using the displaced risk function. In this case, the probability of consider known states as unknown states slightly affects the performance of PR-SRL, which is more evident in PR-SRL $k = 3$, where this probability is higher. The values $k = 3$ and $k = 6$ does not generate failures, but as the sigmoid function begins to look like the discrete function used by the PI-SRL algorithm ($k = 12$ and $k = 48$), the failures appear.



(a)



(b)

Figure 3: (a) Mean cumulative reward per episode obtained by PI-SRL and by PR-SRL using different values of k for nondisplaced sigmoid functions. (b) Mean cumulative reward per episode obtained by PI-SRL and by PR-SRL using different values of k for displaced sigmoid functions. The means have been computed from 10 different executions.

We also have used PR-SRL for improving in a simulated environment the default walking pattern provided by the Nao Robot avoiding falls. In our walking problem, the state and action spaces are continuous. The state space is represented by a 3-dimensional tuple, $< x, y, \theta >$, composed of the lengthwise separation in meters of a feet with the opposite, x, the crosswise separation, y, and a rotational component in the vertical axis, z in radians, θ. The action space is a 4-dimensional tuple, $< \Delta x, \Delta y, \Delta\theta, t >$, that represents the displacement that performs a foot relative to the opposite one for each state variable, and the time used to make such displacement, t. In these experiments, PR-SRL uses the default pre-configured walking pattern as baseline behavior, π_T. Figure 4(b) shows two learning process with different risk configurations and, for each one, the cumulative reward obtained per episode. In this case, the high level of risk ($\sigma = 9 \times 10^{-3}$) obtains a better policy than the low level of risk ($\sigma = 9 \times 10^{-4}$). However, both risk configurations outperform the default pre-configured walking pattern. Figure 4(a) shows the same evaluation but using a tuned (faster) baseline behavior. We show that a better baseline produces better learned behavior (note y axis have different ranges). In this case, the Nao Robot never falls during the learning process.

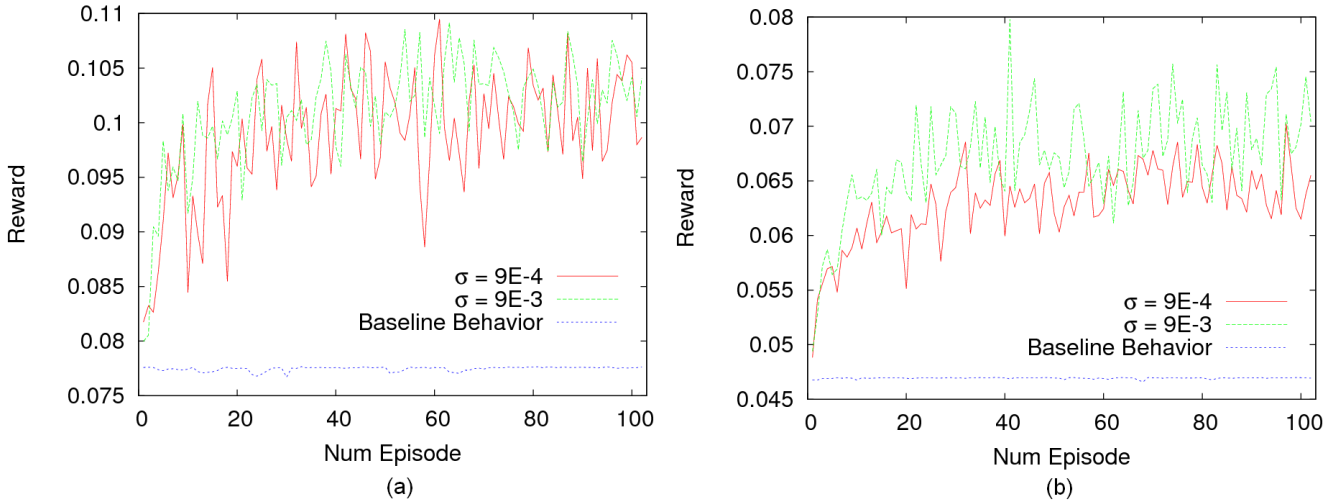


Figure 4: (a) Mean cumulative reward per episode for different risk configurations (σ) using the fastest pre-configured walking pattern as baseline behaviour. (b) Mean cumulative reward per episode for different risk configurations (σ) using the default pre-configured walking pattern as baseline behaviour. The means have been computed from 5 different executions.

4 Conclusions

In this work, we have introduced the Policy Reuse for Safe Reinforcement Learning algorithm. It is based on the introduction of a safe teacher advice in a new learning process. The π-reuse exploration strategy is used to balance exploration, exploitation, and the advice transference. The three elements are weighted by the new risk function, that defines a probability to follow the advice, and the risk parameter, σ that defines the level of randomness (or noise) in action execution. The experiments have demonstrated its suitability in risky domains, where fails must be avoided. Current work is oriented to test the approach in real robotic environments instead of simulated ones.

References

[1] F. Fernández and M. Veloso. Learning domain structure through probabilistic policy reuse in reinforcement learning. *Progress in Artificial Intelligence*, 2(1):13–27, 2013.

[2] J. García and F. Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, Dec. 2012.

[3] P. Geibel and F. Wysotzki. Risk-sensitive Reinforcement Learning Applied to Control under Constraints. *Journal of Artificial Intelligence Research (JAIR)*, 24:81–108, 2005.

[4] J. A. Martín H. and J. Lope. Learning Autonomous Helicopter Flight with Evolutionary Reinforcement Learning. In *12th International Conference on Computer Aided Systems Theory (EUROCAST)*, pages 75–82, 2009.

[5] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry. Autonomous Helicopter Flight via Reinforcement Learning. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *NIPS*. MIT Press, 2003.

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

[7] M. E. Taylor, H. B. Suay, , and S. Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *Proceedings International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[8] L. Torrey and M. E. Taylor. Help an agent out: Student/teacher learning in sequential decision tasks. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-12)*, June 2012.

Lifetime Value Marketing Using Reinforcement Learning

Georgios Theocharous
Adobe Research
theochar@adobe.com

Assaf Hallak
Technion
Adobe Research
ifogph@gmail.com

Abstract

In many marketing applications, companies use technology for interacting with their customers and making product or services recommendations. Today, these marketing decisions are mainly made in a myopic (best opportunity right now) approach and optimize short-term gains. In our research we are exploring new ways of marketing interactions for optimizing Life-Time Value (LTV). In particular, we are exploring marketing recommendations through Reinforcement Learning (RL) and Markov Decision Processes (MDPs). In this paper we compute the LTV policies for several real world data sets using various state of the art reinforcement learning algorithms. In addition, we propose an offline evaluation method for these methods using a well-crafted simulator, according to which LTV policies outperform myopic policies. Finally, we characterize the error of the estimated value of the policies on the simulator, using the simulator's prediction errors.

Keywords: Reinforcement learning, marketing, lifetime value modeling.

1 Introduction

In many marketing application, a company or organization uses technology for interacting with their end customers and making recommendations. For example, a department store might offer customers discount coupons or promotions; an online store might serve targeted "on sale now" offers; or a bank might email appropriate customers new loan or mortgage offers. Today, these marketing decisions are mainly made in a myopic (best opportunity right now) approach and optimize short-term gains. In our research we are exploring new ways of marketing interactions for optimizing Life-Time Value (LTV). LTV can be thought of as long-term objectives such as revenue, customer satisfaction, or customer loyalty. These long-term objectives can be represented as the sum of an appropriate reward function.

These sums of rewards can be computed through a stream of interactions between the company and each customer, including both actions from the company (e.g. promotions, advertisements, or emails) and actions by the customer (e.g. purchases, clicks on a website, or signing up for a newsletter). In our work we are proposing technologies for computing interaction strategies by the company that maximizes the sum of rewards. In particular, we are exploring Reinforcement Learning (RL) and Markov Decision Processes (MDPs) - powerful paradigms for sequential decision-making under uncertainty. In RL problems, an agent interacts with a dynamic, stochastic, and incompletely known environment, with the goal of finding an action-selection strategy, or policy, to maximize some measure of its long-term performance.

In our RL formulation for marketing the agent is an algorithm that takes actions such as showing an ad and offering a promotion; the environment can be thought of as features about customer demographics, the web content and customer's behaviors such as recency (last time the webpage was visited), frequency (how often the page has been visited), and monetary (how much was spent so far); the reward can be thought of as the price of products purchased by the customer in response to an action taken by the marketing algorithm; finally, the goal of the marketing agent is to maximize its long-term revenue.

Using MDPs and RL to develop algorithms for LTV marketing is still in its infancy. Related work has used toy examples and has appeared mostly in marketing venues [11] [3] [7]. An approached directly related to our work first appeared in [6] where the authors used public data of an email charity campaign, and showed that RL policies produce better results than myopic. The authors there used off-line batch RL methods and heuristic simulators for evaluation. Recently in [9] the authors proposed an on-line RL system which learns concurrently from multiple customers. The system was trained and tested on a simulator.

Unlike previous work, we deal with real and big data problems, where we are faced with both the challenge of learning RL policies from high dimensional problems as well as evaluating them off-line. In the rest of the paper we describe our experiments on a few real data sets, where our results show that RL policies are better than myopic. Our evaluation was done on a simulator, which we build using system identification techniques. We also propose theory to characterize the error of the estimated value of the policies on the simulator, given the simulator's error in predicting the dynamics and the rewards.

2 Data and Algorithms

Our data sets originated from various companies from the automotive and banking industry. On the company websites when customers visit, they are shown one of a finite number of offers. The reward is one when a user clicks on the offer and zero otherwise. For every company we extracted/created features as shown in Table 1. The data is highly unbalanced and often has disproportional number of data points for the different offers. To mitigate the problem in our experiments we merge actions together by scoring them according to immediate reward and then clustering them using K-means.

We used a few state of the art Reinforcement learning algorithms, which are able to handle high dimensional continuous variables. A few of them are model-based, where various quantization techniques are applied on the data and for each a state space is obtained. For these methods the transition probabilities and rewards are then inferred from the data, and the optimal policy and its value can be found using standard MDP methods. Below we describe all the algorithms we used:

KBRL-RS Kernel Based Reinforcement learning on Representative States (KBRL-RS) [4], selects the representative states of the data using a *cover tree* [1]. The algorithm then applies Kernel-Based RL on top of the representative states and in effect shows that this is equivalent to learning and solving an MDP on top of the representative states. A new state interpolates its value from the representative states according to the kernel.

K-means-RL K-means-RL creates representative states using the K-means algorithm [5]. It then learns an MDP on top of those states and solves it. New observations are labeled and assigned the actions of their nearest states.

Cum action	There is one variable for each offer, which counts number of times each offer was shown
Visit time recency	Time since last visit
Cum success	Sum of previous reward
Visit	The number of visits so far
Success recency	The last time there was positive reward
Longitude	Geographic location [Degrees]
Latitude	Geographic location [Degrees]
Day of week	Any of the 7 days
User hour	Any of the 24 hours
Local hour	Any of the 24 hours
User hour type	Any of weekday-free, weekday-busy, weekend
Operating system	Any of unknown, windows, mac, linux
Interests	There a finite number of interests for each company. Each interest is a variable that gets a score according to the content of areas visited within the company websites
Demographics	There are many variables in this category such as age, income, home value...

Table 1: Features

FQI Fitted Q iteration (FQI) [2] is a practical RL algorithm. The method is a variation of Q iteration, where the exact Q function is computed by function approximation. The approximation is fitted by a non-parametric regressor, such as ensembles of regression trees. A new state uses the regressor for each action to estimate its Q value.

FQI-sarsa Fitted Q iteration sarsa is the same as FQI but with sarsa style updates. This algorithm learns the value of the behavior policy that produced the data. During execution though the algorithm choses the action with maximum Q value and in effect does one step improvement from the behavior policy.

3 Experiments

A major issue when considering the dynamics of the problem is evaluating a policy offline, i.e. without seeing it in practice. This matter can be solved with importance sampling [8], or when using linear function approximation [10]. However, in practice these methods are limited.

We devise a different approach, fitted for data sets of similar nature to ours - construct a simulator of the dynamics using careful inspection of each feature. The main idea is that if the simulator is similar enough to the true dynamics, the value function of any policy will be close as well to its actual value.

Our simulator is constructed as follows: First we recognize the easy to predict features. These can be static features such as geographic location and income, or other simple features such as cumulative action counter. Next, we identify features that act stochastically and have high impact on other features, such as the time difference and the reward; these are simulated using random sampling from their distribution. All remaining features are predicted using regression trees on the previous observation and the sampled stochastic features.

We performed various sanity checks to test the quality of the simulator such as measuring the error of one step predictions and in examining the return of the behavior policy seen in the data. We also observed that when a policy and the simulator are trained on the same data set, then as the number of states is increased in the policy estimation, using an algorithm such as K-means-RL, so does the performance between the optimal and the myopic policy. This is evidence that the simulator is behaving in a similar overfitting fashion as the underlying K-means MDP used to compute a policy.

We experimented with three big data sets of approximately 600000 interaction each. To avoid overfitting we split each of the data sets into two parts. In the first part we compute the policies and in the second part we use the data to learn a simulator. To obtain error bars we do the data splitting multiple times. For each data set we split it 10 times and run 500 experiments of maximum length 100. We record the average reward and the standard error shown in the parenthesis in Table 2. We compared the optimal strategies with myopic and random. LTV strategies were produced by setting $\gamma = 0.99$, while myopic strategies were produced by setting $\gamma = 0$; in the table we show the average reward per time step. The results in Table 2 highlighted in bold, show that optimal RL policies are better than myopic and random strategies.

4 Simulation Error Characterization

It seems intuitive that a simulator for which the rewards and dynamics are close to the actual system will yield similar values to the actual system for various policies, we can show this is correct in theory. To ease the notation we insert the

Data set	KBRL-RS optimal	K-means -RL optimal	FQI optimal	FQI-sarsa optimal	KBRL-RS myopic	K-means-RL myopic	FQI myopic	Random
1	**0.054** (0.01)	**0.06** (0.01)	0.007 (0.01)	0.015 (0.002)	0.037 (0.006)	0.03 (0.007)	0.044 (0.01)	0.033 (0.01)
2	**0.038** (0.003)	0.03 (0.003)	**0.04** (0.003)	**0.04** (0.002)	0.036 (0.002)	0.02 (0.002)	0.026 (0.005)	0.033 (0.003)
3	0.003 (10^{-4})	0.003 (10^{-4})	0.003 (10^{-4})	**0.004** (10^{-4})	0.003 (10^{-4})	0.002 (10^{-4})	0.002 (10^{-4})	0.002 (10^{-4})

Table 2: Results

policy inside the function itself. Lets assume the true dynamics are given by:

$$x_{t+1} = f(x_t); \quad r_t = g(x_t); \quad V(x_0) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[g(f^t(x_0))],$$

where f^t is the t'th functional power. However, we falsely believe the dynamics are:

$$x_{t+1} = \hat{f}(x_t); \quad r_t = \hat{g}(x_t); \quad \hat{V}(x_0) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\hat{g}(\hat{f}^t(x_0))]$$

Assume the following bounds:

$$\forall x, y : |\mathbb{E}[g(x)] - \mathbb{E}[\hat{g}(y)]| \leq \alpha d(x, y) + \delta,$$
$$\forall x, y : d(f(x), \hat{f}(y)) \leq \beta d(x, y) + \epsilon,$$

where d is some distance function and $\alpha, \beta, \delta, \epsilon$ are positive numbers such that $\beta \cdot \gamma < 1$. These assumptions embody the idea that even for different initial points, the rewards and transitions cannot be too far apart considering the original distance, this way we can guarantee iteratively the rewards are close as well.

When these assumption are true the following is correct:

$$\forall x_0 : \|V(x_0) - \hat{V}(x_0)\| \leq \frac{\alpha\epsilon\gamma}{(1-\gamma)(1-\beta\gamma)} + \frac{\delta}{1-\gamma}. \tag{1}$$

Proof: First observe that:

$$\begin{aligned} d(f^t(x), \hat{f}^t(y)) &\leq \beta d(f^{t-1}(x), \hat{f}^{t-1}(y)) + \epsilon \\ &\leq \beta(\beta d(f^{t-2}(x), \hat{f}^{t-2}(y)) + \epsilon) + \epsilon \\ &... \leq \frac{1-\beta^t}{1-\beta}\epsilon + \beta^t d(x, y) \end{aligned}$$

Now we can bound the value difference as follows:

$$\begin{aligned} |V(x_0) - \hat{V}(x_0)| &\leq \sum_{t=0}^{\infty} \gamma^t |\mathbb{E}[g(f^t(x_0))] - \mathbb{E}[\hat{g}(\hat{f}^t(x_0))]| \\ &\leq \frac{\delta}{1-\gamma} + \sum_{t=0}^{\infty} \gamma^t \alpha d(f^t(x_0), \hat{f}^t(x_0)) \\ &\leq \frac{\delta}{1-\gamma} + \sum_{t=0}^{\infty} \gamma^t \alpha\epsilon \frac{1-\beta^t}{1-\beta} \\ \text{[demand } \beta \cdot \gamma < 1 \text{ for convergence]} &= \frac{\delta}{1-\gamma} + \frac{\alpha\epsilon\gamma}{(1-\gamma)(1-\beta\gamma)} \end{aligned}$$

□

The parameters in the bound can be estimated empirically depending on the distance d used. For example, by using $d(x, y) = \mathbb{E}\|x - y\|$ and the triangle inequality, the following inequalities yield easy-to-find bounds:

$$|\mathbb{E}[g(x)] - \mathbb{E}[\hat{g}(y)]| \leq |\mathbb{E}[g(x)] - \mathbb{E}[\hat{g}(x)]| + |\mathbb{E}[\hat{g}(x)] - \mathbb{E}[\hat{g}(y)]|$$
$$\mathbb{E}\|f(x) - \hat{f}(y)\| \leq \mathbb{E}\|f(x) - \hat{f}(x)\| + \mathbb{E}\|\hat{f}(x) - \hat{f}(y)\|$$

The right hand side of these inequalities can be related to as the variance (first term) and bias (second term) of the model. The first term in the right hand side of the inequalities can be bounded using classical bounds, depending on any previous assumptions on g and f. The second term depends on our own choice of model and therefore can be bounded by construction or empirically. We have yet to find this bound on our data.

5 Conclusions

In our paper we tried to tackle the marketing problem using the less established LTV approach. Although this approach results in a richer representation of the communication with the customer, it is often disregarded due to its assumed higher complexity and the difficulty in evaluating the resulting strategies without expensive testing.

To prove the legitimacy of this approach, we implemented several state of the art algorithms and applied them on the data to obtain LTV maximizing policies. In order to evaluate these offline, we developed a novel simulator based offline evaluation method. In addition, we developed a simple bound that justifies this method, and can be estimated empirically.

Our estimations of the myopic policies versus the LTV optimal policies showed a clear advantage for the latter, but these results are still preliminary. In the future we plan to test our paradigm on more data sets, estimate the simulator's error, optimize the parameters for the planning algorithms and visualize the LTV policies.

References

[1] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104, 2006.

[2] Damien Ernst, Pierre Geurts, Louis Wehenkel, and L. Littman. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[3] Jedid-Jah Jonker, Nanda Piersma, and Dirk Van den Poel. Joint optimization of customer segmentation and marketing policy to maximize long-term profitability. *Expert Systems with Applications*, 27(2):159 – 168, 2004.

[4] Branislav Kveton and Georgios Theocharous. Kernel-based reinforcement learning on representative states. In *AAAI*, 2012.

[5] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[6] Edwin Pednault, Naoki Abe, and Bianca Zadrozny. Sequential cost-sensitive decision making with reinforcement learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 259–268, New York, NY, USA, 2002. ACM.

[7] Phillip E. Pfeifer and Robert L. Carraway. Modeling customer relationships as markov chains. *Journal of interactive marketing*, pages 43–55, 2000.

[8] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[9] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall. Concurrent reinforcement learning from customer interactions. In *In 30th International Conference on Machine Learning*, 2013.

[10] Richard S Sutton, Csaba Szepesvári, and Hamid R Maei. A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems 21*, volume 21, 2008.

[11] Giuliano Tirenni, Abderrahim Labbi, Cesar Berrospi, Andr Elisseeff, Timir Bhose, Kari Pauro, and Seppo Poyhonen. The 2005 isms practice prize winnercustomer equity and lifetime management (celm) finnair case study. *Marketing Science*, 26:553–565, 2007.

Off-Policy Reinforcement Learning with Gaussian Processes

Girish Chowdhary*
School of Mechanical and Aerospace Engineering
Oklahoma State University
Stillwater, OK 74078
girish.chowdhary@okstate.edu

Miao Liu
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708
miao.liu@duke.edu

Robert C. Grande
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
rgrande@mit.edu

Thomas J. Walsh
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139
twalsh@mit.edu

Jonathan P. How
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
jhow@mit.edu

Abstract

An off-policy Bayesian nonparameteric approximate reinforcement learning framework, termed as GPQ, that employs a Gaussian Processes (GP) model of the value (Q) function is presented in both the batch and online settings. Sufficient conditions on GP hyperparameter selection are established to guarantee convergence of off-policy GPQ in the batch setting, and theoretical and practical extensions are provided for the online case. In particular, the convergence results in the batch case extend theoretical results on the Fitted Q-Iteration family of algorithms and the online results provide a theoretical grounding for the use of sparse, budgeted GP representations. These results reveal a reason for potential divergence of off-policy approximate reinforcement learning employing Gaussian kernels as well as hyperparameter selection conditions to eliminate this possibility. Empirical results demonstrate GPQ has competitive learning speeds in addition to its convergence guarantees and its ability to automatically choose its own basis locations.

Keywords: Reinforcement Learning, Gaussian Processes

Acknowledgements

This work was supported in part by the Aerojet Fellowship as well as the Office of Naval Research under MURI program award #N000141110688 and contract #N000140910625.

*A longer version of this work with complete proofs is currently in submission at the Conferene on Neural Information Processing Systems (NIPS-13).

1 Introduction

Reinforcement learning (RL) [17] in continuous or large state spaces often relies on function approximation to maintain a compact model of the value (Q) function [4, 5, 10], but often requires a precise choice of features to achieve good performance. Gaussian Processes (GPs) [15] are Bayesian Nonparametric (BNP) models that are capable of automatically adjusting features based on the observed data and have been successfully employed in high-dimensional approximate RL domains [4], but several properties of RL with GPs, particularly convergence guarantees, off-policy learning, and exploration techniques have not been fully addressed.

More specifically, no convergence results for RL algorithms with GPs exist, and existing RL methods with GPs either require a planner [14, 3, 7] or are restricted to on-policy learning [4]. The latter approach is less general than off-policy RL, which enables learning the optimal value function using samples collected with a safe or exploratory policy. In addition, existing model-free GP RL methods (e.g. [4]) do not fully leverage the Bayesian notion of predictive variance inherent to a GP model in guiding the exploration strategy.

In this abstract, the three problems mentioned above for approximate RL using GPs are addressed: convergence, off-policy learning, and exploration. More specifically, a model-free off-policy approximate reinforcement learning technique, termed as GPQ, that uses a GP model to approximate the value function is presented. Sufficient conditions for convergence of GPQ to the best achievable optimal Q-function given the data (Q^*) are presented in the batch and online settings, and it is shown that these properties hold even as features are added or removed to maintain computational feasibility. We also present an exploration scheme baed on the GPs explicit confidence intervals over Q^*. Unlike other recent papers on off-policy RL with fixed-parameter linear function approximation [18, 10, 8, 9], our approach allows the basis functions to be automatically identified from data. Furthermore, our condition for convergence reveals why GPQ or kernel base Fitted Q-Iteration [6] could diverge, and how this divergence can be prevented by tuning a regularization-like parameter of the GP. More broadly, this work helps to connect the fields of BNP modeling and RL by combining BNP strengths such as prior distributions and data-driven basis construction with RL architectures such as Fitted Q-Iteration [5] in a theoretically grounded manner.

2 Background

We consider domains modeled as a Markov Decision Process (MDP) [17] with standard notation: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, policy $\pi : S \mapsto A$ and optimal value (Q-)function $Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r(s_t, a_t) + \max_{a'} \gamma Q^*(s_{t+1}, a')]$. Reinforcement learning is concerned with finding the optimal policy $\pi^*(s) = \text{argmax}_a Q^*(s, a)$ when $\mathcal{P}$ and $\mathcal{R}$ are unknown. When an RL method learns the value function of the same policy with which samples were collected it is classified as *on-policy*; when the policy employed to obtain samples is different, it is termed *off-policy*. For continuous RL domains, linear value function approximation is often used to model the Q-function as a weighted combination of fixed bases $\phi(s, a)$, that is $Q(s_t, a_t) = \phi^T(s_t, a_t)\theta$. However, function approximation can cause divergence for off-policy RL methods [1, 19].

Gaussian Processes (GPs) [15] are BNP function approximation models: they do not specify a model structure and explicitly model noise and uncertainty. A GP is defined as a collection of random variables, any finite subset of which has a joint Gaussian distribution with mean (prediction) function $m(z)$ and covariance kernel $k(z', z)$, for input points z and z' ($z = \langle s, a \rangle$ in our case). Define $K(Z, Z)$ as the kernel matrix with entries $K_{l,m} = k(z_l, z_m)$. $\mathbf{k}(Z, z_{\tau+1}) \in \mathbb{R}^\tau$ denotes the kernel vector corresponding to the $\tau + 1^{th}$ measurement, and ω_n^2 represents the variance of the measurement uncertainty. The conditional probability can be calculated as a normal variable [15] with mean $m(z_{\tau+1}) = \alpha^T \mathbf{k}(Z, z_{\tau+1})$, where $\alpha = [K(Z, Z) + \omega_n^2 I]^{-1} \vec{y}$ are the kernel weights, and covariance

$$\Sigma(z_{\tau+1}) = k(z_{\tau+1}, z_{\tau+1}) + \omega_n^2 - \mathbf{k}^T(Z, z_{\tau+1})[K(Z, Z) + \omega_n^2 I]^{-1} \mathbf{k}(Z, z_{\tau+1}) \tag{1}$$

In this paper, we use the sparsification method of [2], which allows for sequential updates. This sparsification algorithm works by building a dictionary of basis vector points. In order to determine when a new point should be added to the dictionary, a linear independence test is performed:

$$\beta_{\tau+1} = k(z_{\tau+1}, z_{\tau+1}) - \mathbf{k}(Z_d, z_{\tau+1})^T K(Z_d, Z_d)^{-1} \mathbf{k}(Z_d, z_{\tau+1}). \tag{2}$$

When $\beta_{\tau+1}$ is larger than a specified threshold β_{tol}, then a new data point is added to the dictionary. Otherwise, the weights α_τ are updated, but the dimensionality of α_τ remains the same.

GPs have previously been used in *on-policy* model-free RL [4] and also model-based RL [14, 3, 7]. For linear fixed-basis (non-GP) value function approximation, several recent approaches ensure convergence in the online case by adding some form of regularization to TD algorithms, as done in GQ [10], LARS-TD [8], and RO-TD [9]. In contrast to these algorithms, GPQ attempts to directly minimize the Q-learning TD error and does not require a priori basis function placement. Other algorithms, such as Bellman Error Basis Functions [13], dynamically construct features for linear function approximators, but are primarily used in policy evaluation. In the batch case, several algorithms belonging to the Fitted Q-Iteration (FQI)

Algorithm 1 Batch GPQ (GP-FQI)

Input: Experience tuples $\langle s, a, r, s' \rangle_{1...N}$
Output: A GP representing $\hat{Q}^*$
$\hat{Q} \leftarrow$ Initialized GP.
repeat
  $\hat{Q}' \leftarrow$ Initialized GP.
  for each experience tuple $\langle s, a, r, s' \rangle_i$ **do**
    $y_i = r_i + \gamma \max_b \hat{Q}(s', b)$
  Train $\hat{Q}$ on all $(\langle s_i, a_i \rangle, y)_i$
  $\hat{Q} = \hat{Q}'$
until The convergence condition is satisfied

Algorithm 2 Online GPQ

for for each time step τ **do**
  Choose a_τ from s_τ, using ϵ-greedy exploration
  Take action a_τ, observe r_τ, $s_{\tau+1}$
  Let $z_\tau = \langle s, a \rangle$ and $y_\tau = r + \gamma max_b \hat{Q}(s', b)$
  if $\beta_{\tau+1} > \beta_{tol}$ **then**
    Add z_τ to the $\mathcal{BV}$ set.
  Compute $\mathbf{k}_{z_{\tau+1}}$ and $\alpha_{\tau+1}$ according to [2]
  if $|\mathcal{BV}| >$ Budget **then**
    Delete $z_i \in \mathcal{BV}$ with lowest score according to [2]
  update $\hat{Q}(z_{\tau+1}) = \sum_{i=1}^{\infty} \alpha_i k(z_i, \cdot)$

family of algorithms [5] can guarantee off-policy convergence with specific function approximators, including averagers [12] and certain forms of regularized least squares [6].

3 Off-Policy RL with a GP

Our goal is to perform posterior inference using available information so that the current estimate of the mean $\hat{m}$ approaches the mean of Q^*. Let the current estimate of the mean of the Q-function be $\hat{Q}(s, a) = \hat{m}(s, a)$. Since samples of Q^* are not available, posterior inference needs to be performed using the best estimate of Q^* at the current time as:

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a'}(\hat{Q}(s_{t+1}, a')). \tag{3}$$

Whenever we update the model of $\hat{Q}(s_t, a_t)$ with a new observation, the accuracy of the observation is dependent on the accuracy of the current model. Typically, the parameter ω_n^2 is viewed as a uncorrelated, Gaussian measurement noise in GP literature. Here, we offer an alternative interpretation of ω_n^2 as a regularization term, which accounts for the fact that current measurements are not necessarily drawn from the true model and therefore prevents our model from converging too quickly to an incorrect estimate of Q^*. As we show later, ω_n^2 plays a pivotal role in preventing divergence as well.

3.1 Batch GP-Fitted Q-Iteration

Using GPs and the update rule in Equation 3 in the batch setting gives us Algorithm 1, which we call GP-FQI because it is a member of the Fitted Q-Iteration [5] family of algorithms. Below, we prove that GP-FQI can diverge if the regularization parameter is not properly set. However, we also prove that for any set of hyperparameters and desired density of data, a proper regularization constant can be determined to ensure convergence. We begin with a counter-example showing divergence in the batch setting if ω_n^2 is insufficient, but show convergence when ω_n^2 is large enough.

Consider a system with three nodes on the real line at locations -1, 0, and 1. At each time step, the agent can move deterministically to any node or remain at its current node. The reward associated with all actions is zero. All algorithms are initialized with $\hat{Q}(z) = 1 \forall z$, $\gamma = 0.9999$, and we use a RBF kernel with bandwidth $\sigma = 1$ in all cases. We consider two settings of the regularization parameter, $\omega_n^2 = 0.1$ and $\omega_n^2 = 1$. Figure 1 shows that when ω_n^2 is set too low, the Bellman operator can produce divergence in the batch setting. If the regularization is set to the higher value, GP-FQI converges. In the following sections, we show that determining the sufficient regularization parameter ω_n^2 depends only on the density of the data and the hyperparameters, not the initialization value of $\hat{Q}$ or γ.

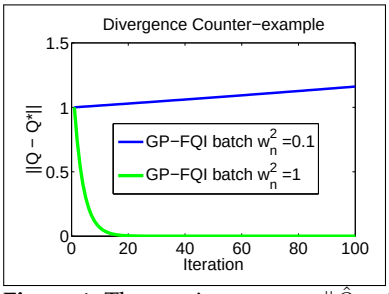


Figure 1: The maximum error $\|\hat{Q} - Q^*\|$ is plotted for GP-FQI with insufficient regularization $\omega_n^2 = 0.1$ and sufficient regularization $\omega_n^2 = 1$.

In the following theorem, we show that in the case of finite data, a finite regularization term always exists which guarantees convergence.

Theorem 1. *Given a GP with data Z of finite size N, and Mercer kernel that is bounded above by $k_{\max}$, there exists a finite regularization parameter ω_n^2 such that the Bellman operator T is a contraction in the batch setting. In particular, $\omega_n^2 = 2(\|K(Z,Z)\|_\infty - k_{\max}) \leq 2N$*

The crux of the proof is showing that the *approximate* Bellman operator is a contraction if the term $\|K(Z,Z)\|\|(K(Z,Z) + \omega_n^2 I)^{-1}\| \leq 1$ when ω_n^2 is set in this way. In the next theorem, we show that for a GP with infinite data that only adds data points which exceed the linear independence test β_{tol}, a finite regularization term also exists.

Theorem 2. *Given a GP with infinite data generated using a sparse approximation with acceptance tolerance β_{tol}, and given a Mercer kernel function that decays exponentially, there exists a finite regularization parameter ω_n^2 such that the Bellman operator T is a contraction in the batch setting.*

The key to the proof is that $\|K(Z,Z)\| = \max_j \sum_i k(z_j, z_i)$, which is convergent for an infinite number of data points selected using the linear independence test in (2). Theorem 2 provides a powerful insight into the convergence properties of GPs in the context of the Bellman operator. As the density of basis vectors increases or as the bandwidth of the kernel function grows, corresponding to decreasing β_{tol}, the basis vector weights α_i become increasingly correlated. As the weights become correlated, changing the weight at one basis vector also changes the weights of nearby basis vectors. It is this sharing of weights that can result in divergence, as seen in [1]. Theorem 2 shows that for a given β_{tol} and kernel function, there exists a finite regularization parameter ω_n^2 that will prevent divergence. In the next theorem, we bound the approximation error from using a sparse representation of a GP versus a full GP. The key to the proof is that the maximum error is linear in β_{tol}.

Theorem 3. *If the sparse GP algorithm is used, the error $\|\mathbb{E}[\hat{Q} - Q^*]\|$ is uniformly, ultimately bounded for the Bellman operator.*

3.2 Online Learning and Exploration with GPQ

In theory, GP-FQI provides a method for learning the Q-function online with provable convergence. Specifically, one could employ an ϵ-greedy exploration policy to guarantee sufficient data collection and perform the batch updates after every step. However, because of the computational costs of such a "batch-sequential" algorithm, we now consider a different GPQ algorithm: *Online GPQ* (Algorithm 2). At each step of Online GPQ, an action is taken using a policy π that ensures ergodicity of the induced Markov chain, and the value $y_\tau = r + \gamma max_b \hat{Q}_\tau(s', b)$ at location z_τ is calculated. The sparse online GP algorithm of [2] is used to determine whether or not to add a new basis vector to the active bases set $\mathcal{BV}$ and then the kernel weights are updated. Below we provide a set of sufficient conditions for convergence in the online case, where C_t and K_t^{-1} are positive definite matrices related to the posterior and the prior covariance. The proof uses similar techniques to Theorem 17 of [11], using an ODE representation of α: $\dot{\alpha}(t) = \mathbb{E}_\pi[q_t S_t]$ and then showing that $\alpha \to \alpha^*$ for each active basis set.

Theorem 4. *With an ergodic sampling policy π, for each active basis set, a sufficient condition for convergence of $\hat{m}(z_t) \to m^*(z_t)$ as $t \to \infty$ online GPQ is $\mathbb{E}_\pi\left[C_t \mathbf{k}_t \mathbf{k}_t^T + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^T\right] \geq \gamma \mathbb{E}_\pi\left[C_t \mathbf{k}_t \mathbf{k}_t^\alpha + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^\alpha\right]$, where $\mathbf{k}_t^\alpha \alpha_t = \max_{a'}(\mathbf{k}^T(x_{t+1}, a'))\alpha_t$.*

While ϵ-greedy exploration helps guarantee convergence, in practice it is often inefficient at gathering useful samples. A guiding principle of more sample-efficient RL algorithms (e.g. [16]) is "optimism in the face of uncertainty", specifically using over-estimates of the value function and greedy actions to balance exploration/exploitation. We propose using the upper confidence tails from the GP as an optimistic value function. Specifically, for any point $\langle s, a \rangle$, the GP will report an upper confidence tail of $m(s) + 2\mathbf{\Sigma}(s_{\tau+1})$ where m and $\mathbf{\Sigma}$ are defined in Section 2. We can then modify Online GPQ to use greedy actions with respect to this upper tail and change the GP update to $\hat{Q}(s_i, a_i) = r(s_i) + \gamma \max_a [\hat{Q}(s_{i+1}, a) + 2\mathbf{\Sigma}(s, a)]$ That is, we use the upper tail of the next state's Q-value in the Bellman update to overestimate the value function (reminiscent of Model-Based Interval Estimation [16]). We leave a detailed theoretical analysis of this technique to future work.

4 Empirical Results and Conclusions

We now present experiments with two variants of Online GPQ, one with the ϵ-greedy exploration and the other using the optimistic values mentioned above. Comparisons are made to Q-learning with function approximation and fixed bases (QL-FB), the GQ [10] algorithm with fixed bases, and tabular Q-learning. Our experiments cover three domains, a discrete 5×5 Gridworld, a continuous state Inverted Pendulum, and continuous state Puddle-World. After parameter tuning (including basement placement for GQ and QL-FB) and cross-validation, the policy learned from all these methods for the three domains are evaluated based on discounted cumulative reward averaged over 20 independent runs and are shown in Figure 2.

In the Gridworld experiments, while all of the algorithms find the optimal policy, the GPQ based methods converge must faster by quickly identifying important areas for basis points. We also see that optimistic exploration using the GP's variance is advantageous, as the algorithm very quickly uncovers the optimal policy. In Inverted pendulum, GPQ quickly finds adequate bases and converges to a near optimal policy while GQ requires more samples. Beyond the "best parameter" results shown in this graph, we also observed GPQ methods are more resilient against small quantizations (budgets) because they are able to select their own bases, while GPQ and QL-FB are far more sensitive to the number and placement of bases. In Puddle-World, basis placement is more challenging and GPQ sometimes converges to a cautious (puddle adverse) policy. Multiple lines are shown for QL-FB and GQ, depicting their best and wort case in terms of parameter settings, as they were extremely sensitive to these settings in this domain. While the best case versions of GQ and QL-FB reached better policies than GPQ, in the worst case, their Q-values appear to *diverge*, as illustrated in the final

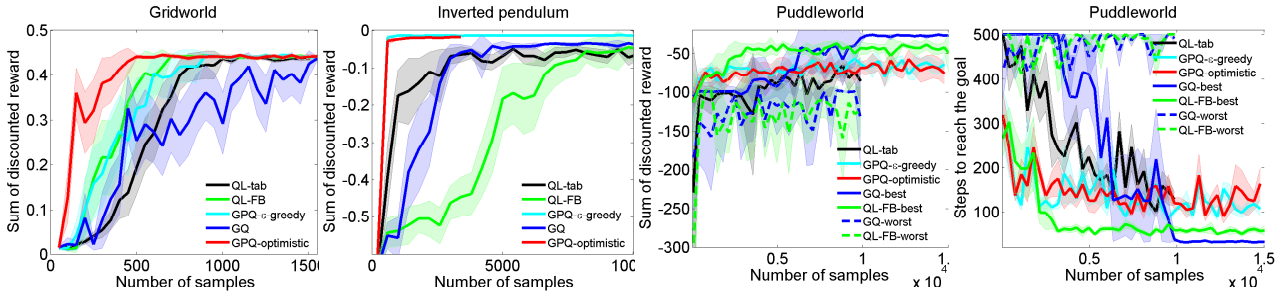


Figure 2: The first 3 figures show the average sum of discounted rewards. The GQ and the QL variants are given more information because their bases are specified a priori, yet GPQ is able to reach comparable performance (often faster) while choosing its own basis functions. The final graph shows the steps to the goal in Puddle World, where the worst-case divergence of GQ and QL-FB is evident.

graph showing the number of steps to the goal, where the worst-case QL-FB and GQ stay near the initial state for all 500 steps. While GQ has convergence guarantees when data comes from a fixed policy, those conditions are violated here, hence the potential for divergence. In summary, while very careful selection of parameters for QL-FB and GQ leads to slightly better performance, GPQ performs almost as well as their best case with less information (since it does not need the bases a priori) and far outperforms their worst-case results.

In this abstract, we presented a NPB framework (GPQ) that uses GPs to approximate the value function in off-policy RL. We presented algorithms using this framework in the batch and online case and provided sufficient conditions for their convergence. Our results show that GPQ's representational power allows it to perform as well or better than other off-policy RL algorithms and that a regularization-like term can help decouple parameters and avoid divergence.

References

[1] L. C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *ICML*, pages 30–37, 1995.

[2] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[3] M. P. Deisenroth. *Efficient reinforcement learning using Gaussian processes*. PhD thesis, Karlsruhe Institute of Technology, 2010.

[4] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *International Conference on Machine Learning (ICML)*, 2005.

[5] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, April 2005.

[6] A. M. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor. Regularized fitted q-iteration for planning in continuous-space markovian decision problems. In *Proceedings of the 2009 American Control Conference*, pages 725–730, 2009.

[7] T. Jung and P. Stone. Gaussian processes for sample efficient reinforcement learning with rmax-like exploration. In *European COnference on Machine Learning (ECML)*, 2012.

[8] J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *ICML*, 2009.

[9] B. Liu, S. Mahadevan, and J. Liu. Regularized off-policy td-learning. In *NIPS*, Lake Tahoe, NV, Dec 2012.

[10] H. R. Maei, C. Szepesvri, S. Bhatnagar, and R. S. Sutton. Toward off-policy learning control with function approximation. In *ICML*, Haifa, Israel, 2010.

[11] F. S. Melo, S. P. Meyn, and M. I. Ribeiro. An analysis of reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*, pages 664–671, 2008.

[12] D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

[13] R. Parr, C. Painter-Wakefield, L. Li, and M. L. Littman. Analyzing feature generation for value-function approximation. In *International Conference on Machine Learning (ICML)*, pages 737–744, 2007.

[14] C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *NIPS*, 2004.

[15] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[16] A. L. Strehl and M. L. Littman. A theoretical analysis of model-based interval estimation. In *ICML*, 2005.

[17] R. Sutton and A. Barto. *Reinforcement Learning, an Introduction*. MIT Press, Cambridge, MA, 1998.

[18] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML*, 2009.

[19] J. N. Tsitsiklis and B. V. Roy. An analysis of temporal difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, May 1997.

Off-Policy Learning Combined with Automatic Feature Expansion for Solving Large MDPs

Alborz Geramifard **Christoph Dann** **Jonathan P. How**

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
77 Massachusetts Ave., Cambridge, MA 02139
{agf,cdann,jhow}@mit.edu

Abstract

Reinforcement learning (RL) techniques with cheap computational complexity and minimal hand-tuning that scale to large problems are highly desired among RL practitioners. Linear function approximation has scaled existing RL techniques to large problems [Lagoudakis and Parr, 2003; Silver *et al.*, 2012], however that technique has two major drawbacks: 1) conventional off-policy techniques such as Q-Learning can be unstable when combined with linear function approximation [Baird, 1995] and 2) finding the "right" set of features for approximation can be challenging.

The first drawback has been recently addressed with the introduction of the Greedy-GQ algorithm, a convergent extension of Q-Learning [Maei *et al.*, 2010]. The second drawback led to representation expansion techniques that add new features along the learning process [Geramifard *et al.*, 2011; Keller *et al.*, 2006; Parr *et al.*, 2007]. Amongst these techniques, incremental Feature Dependency Discovery (iFDD) has shown great potential as it scaled to large problems while enjoying convergence results [Geramifard *et al.*, 2011]. Recently, iFDD$^+$ [Geramifard *et al.*, 2013b] improved the performance of iFDD in the prediction problem while outperforming the previous state-of-the-art batch expansion technique OMP-TD [Painter-Wakefield and Parr, 2012].

This paper connects Greedy-GQ learning with iFDD$^+$ and, for the first time, introduces an online off-policy learning with automatic feature expansion technique. Given sparse features, the new algorithm has per-time-step complexity independent of the total number of features, while for most existing techniques feature discovery is at least quadratic in the number features [Keller *et al.*, 2006; Parr *et al.*, 2007]. Empirical results across 3 domains with sizes up to 77 billion state-action pairs verify the scalability of our new approach.

Keywords: temporal difference learning, representation expansion, off-policy, scaling learning

Acknowledgements

We would like to thank Hamid Maei for his insightful feedback on the use of Greedy-GQ. This research was sponsored by ONR grants N00014-07-1-0749 and N00014-11-1-0688.

1 Introduction

The RL community made a big leap by using linear function approximators to tackle large problems [Lagoudakis and Parr, 2003; Silver *et al.*, 2012]. While using small number of features is much more scalable compared to working with large number of states, it renders off-policy techniques such as Q-Learning unstable [Baird, 1995] and demands careful crafting of the "right" set of features. Off-policy techniques are important part of RL family, as they allow the policy being learnt to be different from the policy used for sampling. To address the latter, Maei *et al.* [2010] recently introduced the Greedy-GQ algorithm as a convergent off-policy technique that employs linear function approximation. To automate the feature crafting process, several representation expansion techniques were developed [Geramifard *et al.*, 2011; Keller *et al.*, 2006; Parr *et al.*, 2007]. Among these methods, the incremental Feature Dependency Discovery (iFDD) [Geramifard *et al.*, 2011] algorithm has shown promise because it successfully scaled to problems with 150 million state-action pairs and has low computational complexity. Recent theoretical analysis on iFDD led to the iFDD$^+$ algorithm that outperformed the previous state-of-the art feature expansion technique in the prediction problem [Geramifard *et al.*, 2013b].

This paper combines Greedy-GQ and iFDD$^+$ algorithms, introducing the first online off-policy technique with automatic feature expansion. Given sparse features, Greedy-GQ-iFDD$^+$ has per-time-step complexity independent of the number of features. Empirical results in domains with up to 77 billion state-action pairs verifies the scalability of our new method.

2 Preliminaries

Markov Decision Processes (MDPs) [Sutton and Barto, 1998] are defined as a tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\mathcal{S}_0,\gamma)$ with $\mathcal{S}$ being the set of all states and $\mathcal{A}$ the finite set of all actions. The transition model $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ specifies the probability of moving from state s to state s' following action a. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the corresponding reward value. The initial state distribution is given by $\mathcal{S}_0 : \mathcal{S} \to [0,1]$. $\gamma \in [0,1[$ is a discount factor. A policy $\pi : \mathcal{S} \to \mathcal{A}$ maps each state to an action. A trajectory is a sequence of s_0, a_0, r_0, s_1, a_1, r_1, $s_2 \ldots$, where $s_0 \sim \mathcal{S}_0$, $a_t = \pi(s_t)$ and $s_{t+1} \sim \mathcal{P}(s_t,a_t,\cdot)$ for $t > 0$. The value of a state-action pair under policy π is defined as the expected cumulative discounted reward (*i.e.*, return) the agent will receive when it takes action a at state s and follows policy π thereafter:

$$Q^\pi(s,a) = \mathbb{E}_{\pi,\mathcal{P}}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right]. \tag{1}$$

An optimal policy π^* maximizes the state-action values, satisfying the Bellman equation:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi^*}(s,a) \quad \forall s \in \mathcal{S}$$

Consequently, calculating Q^{π^*} will be sufficient to find the optimal policy. However this approach requires the storage of $|\mathcal{A}||\mathcal{S}|$ parameters, not amenable for MDPs with large or infinite number of states. Linear function approximation elevates this problem by representing Q^π in a lower dimensional space: $Q_\theta = \theta^T \phi(s,a) \approx Q^\pi$ where $\theta \in \mathbb{R}^n$ is a parameter vector and $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$ is a feature function associating an n-dimensional feature-vector to each state-action pair. For the rest of the paper, we assume $\phi(s,a)$ is formed in two steps. First ϕ maps state s to $\mathbb{R}^m$. Then the resulting vector is copied to the ath slot of a zero vector with $|\mathcal{A}|m = n$ elements, where $|\mathcal{A}|$ is small.

Temporal Difference (TD) learning algorithms improve Q_θ based on the current estimates of Q_θ using sampling techniques. By taking r_0 out of the sum in Equation 1, it can be verified that the value function satisfies

$$Q^\pi(s_t,a_t) = T^\pi Q^\pi(s_t,a_t) \triangleq \mathbb{E}_{\mathcal{P}}[r_t] + \gamma \mathbb{E}_{\mathcal{P},\pi}\left[Q^\pi(s_{t+1},a_{t+1})\right].$$

Hence, the value function is a fixed-point of the *Bellman operator* T^π. The SARSA algorithm [Rummery and Niranjan, 1994] updates the parameters θ_t of the value function estimate at time-step t by $\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi_t$, where α_t is a step-size decreasing over time and δ_t is the TD error of the current transition:

$$\delta_t = r_t + \gamma Q_{\theta_t}(s_{t+1},a_{t+1}) - Q_{\theta_t}(s_t,a_t). \tag{2}$$

The policy π incorporates exploration using the ϵ-greedy policy, where on each step the agent takes a random action with probability ϵ and acts greedy based on the current Q_θ with probability $1 - \epsilon$. Under mild conditions with decaying ϵ, SARSA converges to the optimal policy π^* [Sutton and Barto, 1998]. SARSA is an on-policy technique, because in the TD error calculation (*i.e.*, Equation 2), a_{t+1} is selected according to the current policy π that involves the randomness. Hence, if the random function selects a poor a_{t+1} for which $Q(s_{t+1},a_{t+1})$ has a low value, $Q(s_t,a_t)$ will be panelized accordingly.

Off-policy techniques address this problem by allowing the agent to learn the value function of policy π while collecting samples using a different policy π'. For example Q-Learning [Watkins and Dayan, 1992] can learn about the greedy policy, while collecting samples using an ϵ-greedy policy. Q-Learning updates θ identical to SARSA, yet the TD error is calculated as $\delta_t = r_t + \gamma \operatorname{argmax}_a Q_{\theta_t}(s_{t+1},a) - Q_{\theta_t}(s_t,a_t)$. Unfortunately Q-Learning has been shown to be unstable

with linear function approximation [Baird, 1995]. Recently the Greedy-GQ algorithm [Maei *et al.*, 2010] addressed this drawback by maintaining a second vector ω_t that helps tracking the sub-gradient of the projected Bellman error. The updates for both θ and ω estimates are given by

$$a' = \text{argmax}_a Q_\theta(s_{t+1},a), \tag{3}$$
$$\theta_{t+1} = \theta_t + \alpha_t \left[\delta_t \phi_t - \gamma(\omega_t^\top \phi(s_t,a_t))\phi(s_{t+1},a')\right], \tag{4}$$
$$\omega_{t+1} = \omega_t + \beta_t \left[\delta_t - (\omega_t^\top \phi(s_t,a_t))\right] \phi(s_t,a_t), \tag{5}$$

where δ_t is computed as in Q-Learning and β_t is the learning rate for ω_t. Notice that by setting $\beta_t = 0$ and $\omega_0 = \bar{0}$ we retrieve the Q-Learning algorithm.

Incremental Feature Dependency Discovery (iFDD) [Geramifard *et al.*, 2011] is a recent algorithm for expanding the set of features used for linear function approximation. iFDD assumes the existence of an initial set of binary features ($\boldsymbol{F}$), where $\phi_f : \mathcal{S} \rightarrow \{0,1\}, \forall f \in \boldsymbol{F}$. Through the learning process, iFDD identifies potential features as the conjunction of existing features. For example if $\boldsymbol{F}$ contains 20 features, then a potential feature f can be the conjunction of features 10 and 17, that is $\phi_f(s) = 1 \iff (\phi_{10}(s) = 1 \wedge \phi_{17}(s) = 1)$. Potential features are added as new features once their relevance, η_t, reaches a certain threshold. In the original iFDD algorithm [Geramifard *et al.*, 2011] the relevance of potential feature f at time t is calculated as the cumulative absolute TD error $\eta_t(f) = \sum_{i=0,\phi_f(s_i)=1}^{t} |\delta_i|$. Recently a better relevance criteria was introduced [Geramifard *et al.*, 2013b] based on the rate of convergence analysis of iFDD and its relation to orthogonal matching pursuit algorithms in the prediction case. The new algorithm named, iFDD$^+$, calculates the relevances as

$$\eta_t(f) = \frac{\left|\sum_{i=0,\phi_f(s_i)=1}^{t} \delta_i\right|}{\sqrt{\sum_{i=0,\phi_f(s_i)=1}^{t} 1}}. \tag{6}$$

3 Off-policy Learning with Automatic Feature Expansion

This section combines the Greedy-GQ algorithm with iFDD$^+$ and introduces a novel online off-policy algorithm with feature expansion capability. Algorithm 1 shows the main process. α_t and β_t are the two step size learning parameters (Equations 4-5). ϵ is the exploration used in the ϵ-greedy policy. ξ is the discovery threshold controlling feature expansion. $\boldsymbol{F}$ is the initial set of binary features. ψ and N are two hash maps calculating the sum of TD errors and number of times a feature visited (numerator and denominator of Equation 6). The algorithm follows an ϵ-greedy policy and performs Greedy-GQ updates (Equations 3-5), except for lines 11 and 12. Line 11 executes the *Discover* function defined in Algorithm 2 to expand features. Notice that instead of $\phi(s,a)$, $\phi(s)$ is passed for feature discovery, as all features are shared across all actions. Line 12 pads θ and ω vectors in case of feature expansion. In Algorithm 2, changes to the original iFDD algorithm [Geramifard *et al.*, 2011] to reflect iFDD$^+$ (*i.e.*, Equation 6) are highlighted with the yellow background. All ϕ functions call Algorithm 3 to incorporate new expanded features. Note that, we fixed a drawback of the earlier version of this function [Geramifard *et al.*, 2011] (shown as yellow) which could have led to activation of unnecessary features.

3.1 Per-Time-Step Complexity

Define k_t as the maximum number of non-zero elements for all feature vectors at time t. It can be verified that the per-time-step complexity of Greedy-GQ is $\mathcal{O}(|\mathcal{A}|k_t) = \mathcal{O}(k_t)$. The transition from iFDD to iFDD$^+$ does not raise the per-time-step complexity of the algorithm which is $\mathcal{O}(k_t 2^{k_t})$ [Geramifard *et al.*, 2011]. Hence Greedy-GQ-iFDD$^+$has per-time-step complexity of $\mathcal{O}(k_t 2^{k_t})$. Furthermore, it has been shown that using iFDD, $\forall i > j \rightarrow k_i \leq k_j$ [Geramifard *et al.*, 2011], meaning as new features are discovered the sparsity increases resulting in faster calculations. Due to the exponential term introduced in the complexity of iFDD, the resulting algorithm will be applicable when sparse features are used (*i.e.*, $k_0 \ll m$). Section 4 shows that this condition can be met even in very large domains.

4 Experimental Results

This section investigates the performance of Greedy-GQ (shown as GQ), Q-Learning (*i.e.*, GQ with $\beta_t = 0$), and SARSA with feature expansion (*i.e.*, iFDD$^+$) and without feature expansion (*i.e.*, fixed sparse representation, FSR) across three MDPs: Inverted Pendulum Balancing, BlocksWorld, and Persistent Search and Track. Results are generated using the RLPy framework which is available online [Geramifard *et al.*, 2013a]. For each method 30 runs were used in which after each tenth part of the experiment, a single return of the algorithm was sampled with no exploration. All methods used the same set of random seeds. α_t took the form

$$\alpha_t = \frac{\alpha_0}{k_t} \frac{N_0 + 1}{N_0 + \text{Episode\#}^{1.1}},$$

Algorithm 1: Greedy GQ-iFDD$^+$

Input: α_t,β_t,ϵ,ξ,$\boldsymbol{F}$
Output: Q_θ

Initialize ψ,N to empty maps.
while *time permits* **do**
  initialize s from $\mathcal{S}_0$
  repeat
    $a \leftarrow \epsilon$-greedy w.r.t Q_θ
    s',$r \leftarrow$ execute a
    $a' \leftarrow \text{argmax}_{a'}\, Q_\theta(s',a')$
    $\delta \leftarrow r + Q_\theta(s',a') - Q_\theta(s,a)$
    $\theta \leftarrow \theta + \alpha_t \left[\delta\phi(s,a) - \gamma\left(\omega^\top\phi(s,a)\right)\phi(s',a')\right]$
    $\omega \leftarrow \omega + \beta_t \left[\delta - \phi(s,a)^\top\omega\right]\phi(s,a)$
    $\phi \leftarrow$ Discover$(\phi(s),\delta,\xi,\boldsymbol{F},\psi,N)$
    Pad ω and θ if new features are added.
    $s \leftarrow s'$
  until *s is terminal*;

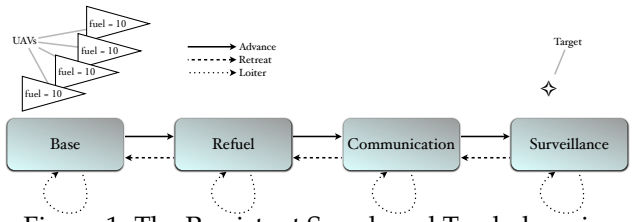


Figure 1: The Persistent Search and Track domain.

Algorithm 2: Discover

Input: $\phi(s)$,δ,ξ,$\boldsymbol{F}$,ψ,N
Output: $\boldsymbol{F}$,ψ

foreach $(g,h) \in \{(i,j)|\phi_i(s)\phi_j(s) = 1\}$ **do**
  $f \leftarrow g \wedge h$
  if $f \notin \boldsymbol{F}$ **then**
    $\psi_f \leftarrow \psi_f + \delta$
    $N_f \leftarrow N_f + 1$
    if $|\psi_f|/\sqrt{N_f} > \xi$ **then**
      $\boldsymbol{F} \leftarrow \boldsymbol{F} \cup f$

Algorithm 3: Generate Feature Vector (ϕ)

Input: $\phi^0(s)$,$\boldsymbol{F}$
Output: $\phi(s)$

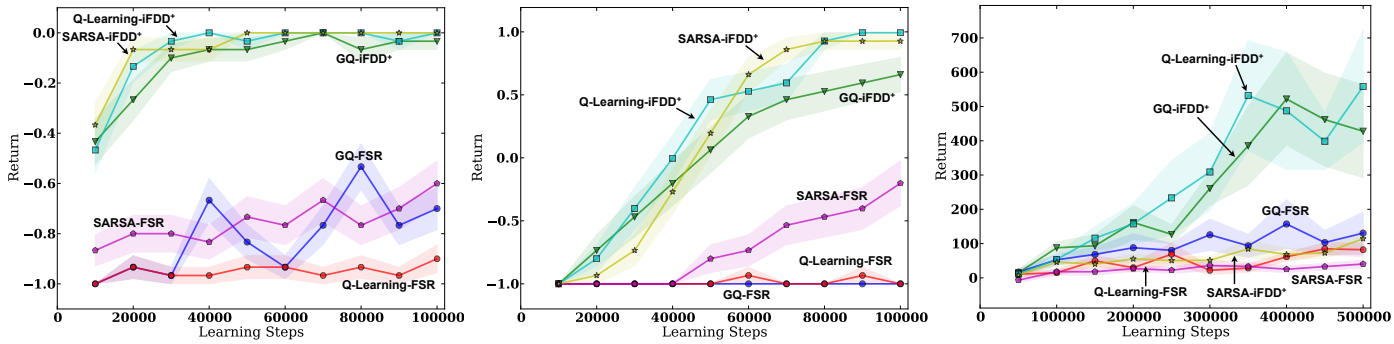
$\phi(s) \leftarrow \overline{0}$
activeInitialFeatures $\leftarrow \{i|\phi_i^0(s) = 1\}$
Candidates $\leftarrow$ SortedPowerSet(*activeInitialFeatures*)
while *activeInitialFeatures* $\neq \emptyset$ **do**
  $f \leftarrow$ *Candidates*.next()
  if $f \subset$ *activeInitialFeatures* **and** $f \in \boldsymbol{F}$ **then**
    activeInitialFeatures $\leftarrow$ *activeInitialFeatures* $\setminus f$
    $\phi_f(s) \leftarrow 1$
return $\phi(s)$

where the best parameters α_0 and N_0 were empirically found from $\{0.1,1\}$ and $\{100,1000,10^6\}$ for each algorithm and domain. Greedy-GQ used $\beta_t = 10^{-6}\alpha_t$ and we set $\epsilon = 0.1$.

Inverted Pendulum Balancing is a widely used benchmark in the control and reinforcement learning community, where a pole mounted on a cart has to be balanced upright by applying force on the cart. We followed the setting of Lagoudakis and Parr [2003] where angle and angular velocity of the pole define the state. The agent can either apply no force or push the cart with $50N$ to the right or left. The pendulum is initialized vertically and the episode stops as soon as the pole reaches the horizontal position with reward -1. All other steps have reward 0. The pendulum is disturbed by uniform noise on the actions between $\pm 10N$. We set $\gamma = 0.95$. Episodes were capped at 3,000 steps. Each state dimension was discretized into 20 bins resulting in 40 initial features per action and k_0 to be 2. This MDP has 1,200 discretized state-action pairs. ξ was empirically found from the set $\{0.1,0.2,0.5\}$.

BlocksWorld is a classical planning problem with the goal of stacking all blocks with a predefined order. We used 6 blocks all initially on the table [Geramifard *et al.*, 2011]. The noise for each movement was 30% resulting in dropping the moving block on the table. The reward is $+1$ for a successful tower build and -0.001 for every other step. The state was defined by a 6 dimensional vector where $s_i = j$ corresponds to block i being on top of block j. For compactness, we interpreted $s_i = i$ for block i being on the table. This MDP has about 1.6×10^6 state-action pairs. Initial features were generated by indicator functions for the position of each block, amounting to 36 features per action and $k_0 = 6$. We set $\gamma = 1$, episodes were capped at 1,000 steps, and ξ was empirically found from the set $\{0.02,0.05,0.1\}$.

Persistent Search and Track (PST) is an MDP with the task of surveilling a target using unmanned aerial vehicles (UAVs) as described in [Geramifard *et al.*, 2011] and shown in Figure 1-(c). Each UAV has three actions $\{retreat,loiter,advance\}$ moving it along the graph. The state of each UAV is described by its $\{fuel,location,motor,camera\}$, where fuel is a discrete value between 0 and 10. The location is the node on the graph. The motor and camera are binary variables indicating the functionality of the corresponding equipment. An additional UAV was added to the previous instantiation of the domain [Geramifard *et al.*, 2011], raising the size of the state-action space to more than 77 billion state-action pairs. The movement of UAVs are deterministic. The motor and the camera of each UAV have 5% chance of failure on each step. The reward of $+20$ is collected when a UAV with a working camera is at the surveillance node and a UAV with a working motor hovers at the communication node. UAVs lose one fuel cell per action and gain full fuel at the refuel node. Both the motor and the camera of a UAV are fixed upon its arrival at base. A scenario is terminated if a UAV crashes due to fuel depletion with a reward of -50. Initial features were generated using indicator functions for each



(a) Inverted Pendulum Balancing (b) Blocksworld (c) Persistent Search and Track

Figure 2: Learning performance in three domains: results are averaged over 30 runs. Shaded areas indicate the standard error of the sampled mean.

dimension of each UAV ignoring zero values for binary dimensions amounting to $17 \times 4 = 68$ features and $k_0 = 16$. We set $\gamma = 0.9$, episodes were capped at 1,000 steps, and ξ was empirically found from the set $\{75{,}100{,}150\}$.

Discussion of Results Figure 2 depicts the simulation results. The Y-axis is the average return of each technique, and the X-axis shows the number of interactions. Shaded areas highlight the standard error of the mean. For the first two problems, the number of interactions were capped to 10^5 steps. For the last domain, due to the large size of the problem, this number was increased to $5{\times}10^5$. In the pendulum domain (Figure 2-(a)), the feature expansion ability enabled all learning techniques to perform close to optimal after 30,000 steps, while no agent with the fixed representation could reliably balance the pendulum by the end of the learning horizon. The same trend can be observed in the BlocksWorld domain (Figure 2-(b)), although one can observe that SARSA-FSR learns better policies compared to off-policy techniques using FSR. Furthermore, among agents using iFDD$^+$, the gap between GQ and Q-Learning has widened. In the largest problem (Figure 2-(c)), among agents using iFDD$^+$, GQ and Q-Learning performed well, while SARSA performed poorly. We suspect this phenomenon is due to higher noise in on-policy TD errors caused by random actions. Off-policy techniques discard the random actions during TD error calculation, providing better criteria for feature expansion. This effect is visible in this domain, because the consequence of random actions can be much more substantial. Agents using FSR performed similar to SARSA-iFDD$^+$ resulting in poor policies. Notice that compared to iFDD, table-lookup representations have been shown to lead to drastic increase in the sample complexity for all three domains [Geramifard *et al.*, 2011].

5 Conclusion

We combined Greedy-GQ algorithm with iFDD$^+$ and introduced the first online off-policy control algorithm with the ability to expand the representation. As shown in Section 3.1, given sparse features, the new algorithm has a per-time-step complexity independent of the total number of features. Empirical results across three domains with sizes up to 77 billion state-action pairs verified the great potential of using off-policy learning with automated feature expansion.

References

Leemon Baird. Residual Algorithms : Reinforcement Learning with Function Approximation. In *International Conference on Machine Learning*, 1995.

Alborz Geramifard, Finale Doshi, Joshua Redding, Nicholas Roy, and Jonathan How. Online discovery of feature dependencies. In Lise Getoor and Tobias Scheffer, editors, *International Conference on Machine Learning (ICML)*, pages 881–888. ACM, June 2011.

Alborz Geramifard, Robert H Klein, and Jonathan P How. RLPy: The Reinforcement Learning Library for Education and Research. http://acl.mit.edu/RLPy, April 2013.

Alborz Geramifard, Thomas J. Walsh, Nicholas Roy, and Jonathan How. Batch iFDD: A Scalable Matching Pursuit Algorithm for Solving MDPs. In *Proceedings of the 29th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Bellevue, Washington, USA, 2013. AUAI Press.

Philipp W. Keller, Shie Mannor, and Doina Precup. Automatic Basis Function Construction for Approximate Dynamic Programming and Reinforcement Learning. In *International Conference on Machine Learning*, 2006.

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.

Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S. Sutton. Toward off-policy learning control with function approximation. In Johannes Fürnkranz and Thorsten Joachims, editors, *International Conference on Machine Learning (ICML)*, pages 719–726. Omnipress, 2010.

Christopher Painter-Wakefield and Ronald Parr. Greedy Algorithms for Sparse Reinforcement Learning. In *International Conference on Machine Learning*, 2012.

Ronald Parr, Christopher Painter-Wakefield, Lihong Li, and Michael Littman. Analyzing Feature Generation for Value-Function Approximation. In *International Conference on Machine Learning*, 2007.

G. A. Rummery and M. Niranjan. Online Q-learning using connectionist systems (tech. rep. no. cued/f-infeng/tr 166). *Cambridge University Engineering Department*, 1994.

David Silver, Richard S. Sutton, and Martin Müller. Temporal-difference search in computer go. *Machine Learning*, 87(2):183–219, 2012.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

Searching for a One-Dimensional Random Walker with Time/Energy Budget

Narges Noori
Department of Computer Science & Engineering
University of Minnesota
Minneapolis, MN, USA
noori@cs.umn.edu

Alessandro Renzaglia
Department of Computer Science & Engineering
University of Minnesota
Minneapolis, MN, USA
arenz@cs.umn.edu

Volkan Isler
Department of Computer Science & Engineering
University of Minnesota
Minneapolis, MN, USA
isler@cs.umn.edu

Abstract

How can we find a target as quickly as possible? This question lies at the core of search theory. Its answer depends on the motion and observation models of the searcher and the target.

We have been studying a classical version of this search problem where the searcher tries to find a target performing a random walk. Our study is motivated by our ongoing field work on using Autonomous Surface Vehicles (ASVs) to find radio-tagged fish in inland lakes. For this problem, it has been observed that the fish are likely to be close to the shore. As a result, its motion can be projected onto a line segment and the search task can be modeled as the problem of finding a one-dimensional random walker. Surprisingly, very little is known about this basic search problem. In our application, an interesting aspect is that the interference between the motor and the radio signal from the fish makes the sensing during the ASV motion unreliable. Therefore, the ASV must briefly stop and listen to the signal to detect the fish. Otherwise, the ASV and the fish can cross each other without the ASV detecting the fish.

The problem can be described as follows. Let L be a set of N discrete locations on the line. We can model the search task using a Markov Decision Process (MDP) with states corresponding to (x, y, t, e) where $x, y \in L$ are the locations of the searcher and the target respectively, t is the current time step and e is the battery level of the searcher. The target is detected if $x = y$. At each time step, the searcher chooses one of the left, right or stay actions and pays the associated energy cost. The target's choice is given by the random-walk model. The goal of the searcher is to choose its actions so as to maximize the capture probability within a time or energy budget. Since the searcher does not know the location of the target unless they are co-located, the variable y is partially observable. The searcher's belief state can now be represented as a tuple (x, Y, t, e) where the new variable Y is a distribution which represents the probability that the target is at a particular location.

In this extended abstract, we provide a summary of our ongoing work. In particular, we show how the POMDP mentioned above can be solved efficiently by binning the distribution into $\log(N)$ cells. This solution provided insights into the structure of optimal solutions, which led us to focus on a class of strategies $(R^k S)^n$. We report our progress toward finding the optimal choice of k and further providing bounds on the quality of this solution.

Our goal is to leverage these insights and to find the optimal strategy for this fundamental search problem first for the one-dimensional case and then in higher dimensions. These strategies can in turn provide insights also for solving more general POMDPs which arise in robotics applications.

Keywords: Pursuit Evasion, Random Walk, POMDP.

Acknowledgements

This work is supported by National Science Foundation Awards #1111638 and #0917676.

Robots are being used in a large number of applications such as surveillance, rescue and environmental monitoring. In many of these cases, the task can be modeled as a pursuit evasion game where the goal of the robot (the pursuer) is to capture a target (the evader). Our motivating application is monitoring the radio-tagged invasive fish using an Autonomous Surface Vehicle (ASV) [1]. Here, the target (fish) is not adversary and since its motion model is unknown we can model it as a simple random walk. Also, since the noise from the motors interferes with the signal from the fish when the ASV is moving, we fix a discrete set of locations for detecting the fish.

At each time step, from any of these locations, the searcher can move to the left, to the right, or stay on its current node. Upon performing each of these actions, the searcher incurs the associated cost, e.g. the energy required for executing the action. These costs can vary along the path, as in the case of a robot equipped with a solar panel [2] that can harvest energy during the task. Therefore, the total cost of an action is the sum of the corresponding loss and gain. The goal is designing search strategies that maximizes the probability of capturing the target given the initial energy budget for the searcher.

In the following, we first present the problem statement and then we explain our POMDP approach for finding the optimal strategy.

1 Problem Statement

The environment is composed of a discrete set of locations $[0, 1, \ldots, N]$ on a line segment. The target starts from an unknown node, and thus we assume that the initial probability distribution of target's possible locations is uniform $\frac{1}{N+1}$. Afterward, the target performs a *simple random walk* as follows. From location $0 < i < N$, with probability q it moves one unit to the *right*, and with the remaining probability $(1 - q)$ it moves one unit to the *left*. We assume that the boundary points 0 and N are reflective (see Fig. 1(a)). Moreover, throughout the paper, we consider a symmetric random walk, i.e. $q = (1 - q) = 0.5$, but our proposed approach works for other values of q as well as for the case that there is a non-zero probability of stay for the target.

The searcher starts from the left-most point $i = 0$. At each time step, it can move to the right or to the left or stay at its current node. Throughout the paper, we refer to these actions as R, L, S respectively. The searcher's strategy is defined as a sequence of these actions, e.g. $R^i S^j L^k$ represents i steps to the right, followed by j stay actions and then k steps to the left.

One definition of capture is when searcher and target are both on the same node at the same time. In this definition, the crossing events such as the one shown in Fig. 1(b) are not considered as capture events [3]. Alternatively, allowing also sensing on edges, these crossing events can be included in the capture events [2].

Finally, the objective is to design the capture strategy $\mathcal{S}$ such that the probability of capture is maximized subject to limitations on either a time budget [3] or an energy budget [2, 4]. In the former case, the cost of all actions is the same and we have the constraint on the total number of actions i.e. T. In the latter case, each action has a different cost and the constraint is on the initial energy of the searcher.

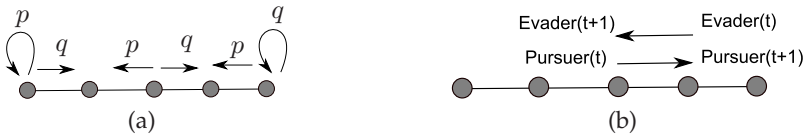


Figure 1: (a) Target's motion model. Here $p = (1 - q)$. (b) The pursuer will miss the target if they cross each other.

2 The POMDP approach

Since the only observation that the searcher has is that it has not captured the target yet, the problem can be formulated as a Partially Observable Markov Decision Process (POMDP) which can be converted to an MDP by including the belief of the searcher as part of the state. By defining the reward function as described in [2], the policy that maximizes the collective reward is in fact the one that maximizes the probability of capture. Fig. 2 depicts the state diagram of this MDP.

The challenge in this formulation is the large number of states. The belief itself is N-dimensional: if we discretize the values in each dimension to k, there will be $O(k^N)$ possible values for the belief which is impossible to track. When crossing is not allowed, the belief is smooth and thus it can be approximated by a specific function which can be represented by a small number of parameters [2]. However, in the crossing case the belief has a jagged pattern and hence the aforementioned approximation approach does not work. See Fig. 3(a) for an example of the belief function when crossing is allowed. In this case, we represent the belief by bins with exponentially increasing width as follows. The i^{th}

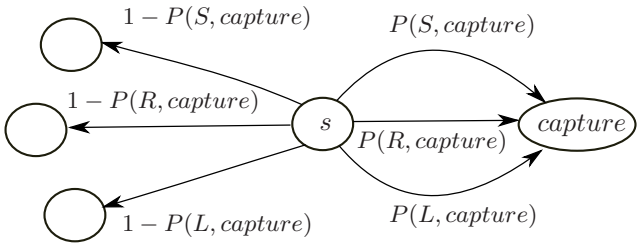


Figure 2: MDP state transitions. In state s, by performing $a \in \{R, L, S\}$, with probability $P(a, capture)$ the searcher captures the target and with the remaining probability the state becomes s'.

bin starts at $c \pm 2^i, 0 \leq i \leq \log(N)$ where c is the current location of the searcher. The approximate belief in each bin is uniform and can be computed as follows. We first compute the cumulative belief in each bin. Then we take the average of this cumulative value in the corresponding bin. Finally, we assign the closest discretization level to this average as the bin value. An example is shown in Fig. 3(b).

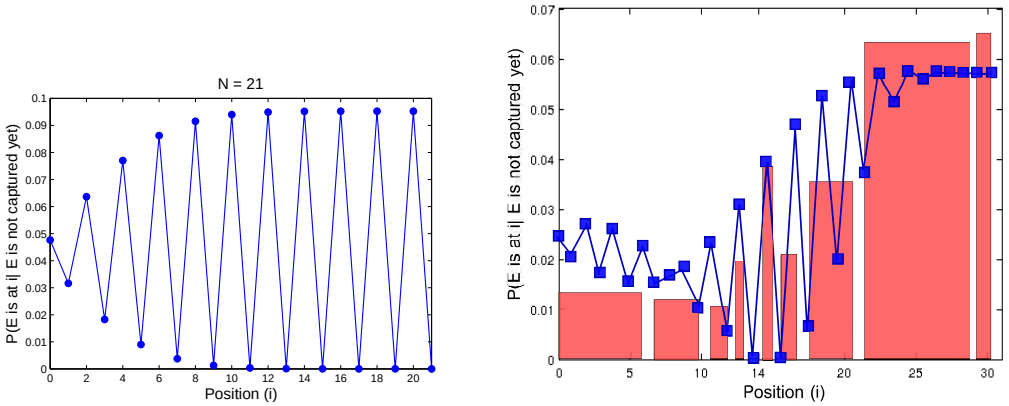


Figure 3: (a) Searcher's belief after a complete sweep i.e. 21 right actions. (b) Belief approximation by bins. The searcher is at $i = 14$.

The solutions obtained from our POMDP formulation suggest that strategies of the form $(R^k S)^m$ are comparable with the POMDP strategies [3]. We show that the capture probability of these strategies can be closely approximated by $\frac{T+m}{2N}$. Furthermore, the maximum probability is achieved for $k = 2$.

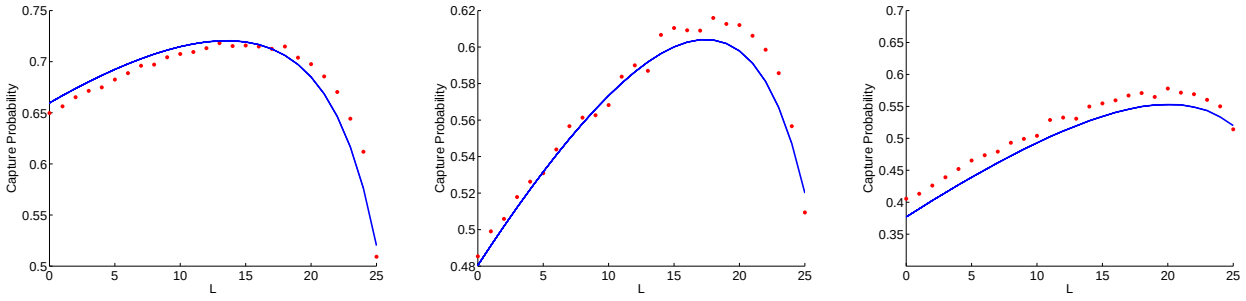
3 Analytical Results

For some determined scenarios, we studied this problem also from an analytical point of view. The main idea is, firstly, to find an analytical approximation for the expected probability of capturing the target given a class of strategies. In [2], where we analyzed the case in which crossing without capture is not possible, we showed that an optimal strategy has to assume a particular structure. For a searcher starting from the left-most point, it is $R^j S^k$, i.e. the searcher moves for j consecutive steps to the right and then keeps the position for k steps. The parameters j and k are constrained by the time/energy constraint. For this class of strategy we showed that the capture probability can be expressed by:

$$P_c \approx \frac{L+1}{N} + \left(1 - \frac{L+1}{N}\right)\left(1 - e^{-\frac{6(E_0 - c_m L)}{c_s[4(N-L-1)^2 - 3(N-L-1)-1]}}\right) \quad (1)$$

where L is the final searcher's location, E_0 is the energy budget and c_m, c_s are the cost for one step move and stay actions respectively. Note that, since the considered function is continuous in the closed and bounded interval $[0, L_{max}]$, where $L_{max} = E_0/c_m$, it always admits a maximum value.

In Figure 4 the comparison between the expected capture probability and the results obtained in simulation is shown. In particular, we considered a segment composed by $N = 50$ nodes. The initial energy budget is $E_0 = 50$ and the cost for moving $c_m = 2$. The three plots presented in Figure 4 correspond to different values of the cost c_s. We see that with these values the solution of the optimization problem is not trivial and the best final searcher position L^* is $0 < L^* < L_{max}$.



(a) (b) (c)

Figure 4: Comparison between the approximations of the expected total capture probabilities (blue line) with the behaviors obtained in simulation (red dots) in function of the swept region L. The values considered are $N = 50$, $E_0 = 50, c_m = 2$ and $c_s = 0.03, 0.05, 0.07$ for (a), (b) and (c) respectively.

For the more complicated case that includes the possibility of crossing without capture, we studied a class of randomized strategies. For such strategies, the stay action occurs with a fixed probability w, which is the optimization parameter. In [4] we obtained an expression similar to (1) for this case.

4 Concluding Remarks

In this work, we study strategies for finding a random walker on a line segment subject to constraints on searching time and energy. We derive analytical solutions for special cases and we propose a POMDP approach that can be used for the general case i.e. with varying costs and gains along the line.

Our future steps are the generalization to two dimensional environments and the relation of our approximate solutions to the optimal strategy.

References

[1] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler, "A robotic system for monitoring carp in minnesota lakes," *Journal of Field Robotics*, vol. 27, no. 6, pp. 779–789, 2010. [Online]. Available: http://dx.doi.org/10.1002/rob.20364

[2] N. Noori, P. Plonski, A. Renzaglia, P. Tokekar, J. Vander Hook, and V. Isler, "Long-term search through energy efficiency and harvesting," Department of Computer Science & Engineering, University of Minnesota, Tech. Rep. 13-001, 2013. [Online]. Available: http://www.cs.umn.edu/research/technical_reports/view/13-001

[3] N. Noori, A. Renzaglia, and V. Isler, "Searching for a one-dimensional random walker: Deterministic strategies with a time budget when crossing is allowed," in *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2013.

[4] A. Renzaglia, N. Noori, and V. Isler, "Searching for a one-dimensional random walker: Randomized strategy with energy budget," in *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2013.

Emergent collective behaviors in a multi-agent reinforcement learning based pedestrian simulation

Francisco Martinez-Gil
Departament d'Informàtica
Universitat de València
Burjassot 46100. Valencia. Spain
Francisco.Martinez-Gil@uv.es

Fernando Fernández
Department of Computer Science
Universidad Carlos III
Campus de Leganés 28911. Madrid. Spain
ffernand@inf.uc3m.es

Miguel Lozano
Departament d'Informàtica
Universitat de València
Burjassot 46100. Valencia. Spain
Miguel.Lozano@uv.es

Abstract

In this work, a Multi-agent Reinforcement Learning framework is used to get plausible simulations of pedestrians groups. In our framework, each virtual agent learns individually and independently to control its velocity inside a virtual environment. The case of study consists on the simulation of the crossing of two groups of embodied virtual agents inside a narrow corridor. This scenario permits us to test if a collective behavior, specifically the lanes formation is produced in our study as occurred in corridors with real pedestrians. The paper studies the influence of different learning algorithms, function approximation approaches, and knowledge transfer mechanisms in the performance of the learned pedestrian behaviors. Specifically, two different RL-based schemas are analyzed. The first one, Iterative Vector Quantization with Q-Learning (ITVQQL) improves iteratively a state-space generalizer based on vector quantization. The second scheme, named TS, uses Tile coding as the generalization method with the Sarsa(λ) algorithm. Knowledge transfer approach is based on the use of Probabilistic Policy Reuse to incorporate previously acquired knowledge in current learning processes; additionally, value function transfer is also used in the ITVQQL schema to transfer the value function between consecutive iterations. The results demonstrate empirically that our RL framework generates individual behaviors capable of emerging the expected collective behavior as occurred in real pedestrians. This collective behavior appears independently of the generalization method used, but depends extremely on whether knowledge transfer was applied or not. In addition, the use of transfer techniques has a notable influence in the final performance (measured in number of times that the task was solved) of the learned behaviors. A video of the simulation is available at the URL: http://www.uv.es/agentes/RL/index.htm

Keywords: Pedestrian Simulation, Transfer Learning, Policy Reuse, Vector Quantization, Tile coding.

Acknowledgements

This work has been partially supported by the University of Valencia under project UV-INV-PRECOMP13-115032, the Spanish Government under project TIN2012-38079-C03-02, the Spanish MICINN under grant TIN2009-14475-C04-04

1 Introduction

The use of Reinforcement Learning (RL) techniques in graphics, animation and simulation tools is gathering increasing attention. Its use has mainly focused on selecting adequate frames from a collection of pre-computed movements or poses to generate interactive animations (Treuille et al. 2007) or to create video textures (Schödl & Essa 2000). In this work, we use RL to control directly the characters of a simulation. Thus, the embodied agents learn to navigate inside a virtual environment to simulate pedestrians groups. Different areas, such as architecture, civil engineering and game development, can benefit from the simulation of pedestrians groups, in order to check the capacities of the facilities in a building, to prevent accidents, or to give realism in urban scenarios. In our framework (Martinez-Gil et al. 2012*b*), each embodied agent learns autonomously to control its velocity to reach to a goal. It uses an Open Dynamics Engine (ODE) calibrated with values of real pedestrians (Martinez-Gil et al. 2012*a*) that simulates the interactions at the physical level.

To design a RL framework with a continuous state space where multiple independent learning processes are carried out at the same time is a challenging problem. Additionally, the pedestrian simulations are particularly difficult because collective behaviors emerge in real pedestrian groups in specific situations like the lane formation in crowded streets. In such situations, it is important to know whether the use of different configurations in the RL framework is critic for the quality of the learned behaviors. In this work, we present a study about the influence of knowledge transfer techniques and the use of different state space generalization methods in the performance of a well-known pedestrian simulation scenario. The narrow-corridor scenario is a problem of pedestrian dynamics where two groups of pedestrians inside a narrow corridor have to cross in order to reach to the opposite side. This scenario is specially suitable to study the emergence of collective behaviors, specifically the lane formation (Helbing et al. 2005). There are two main motivations:

1. To demonstrate empirically that collective behaviors emerge in different configurations of the learning processes.
2. To study the influence of the state space generalization method jointly with a learning algorithm and the use of transfer knowledge techniques in the performance of the learned pedestrian behaviors.

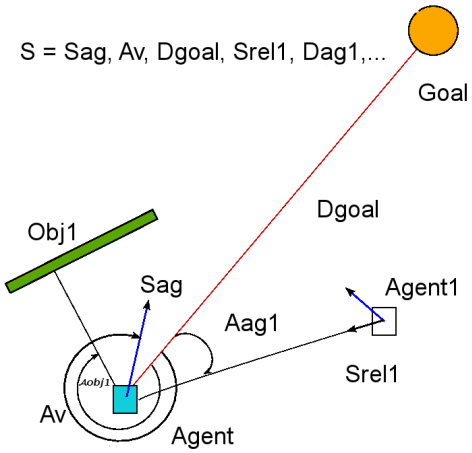
2 The multi-agent RL framework

RL uses optimization techniques to learn from a reward signal a sequential decision-based controller. In RL, the problems are modeled as Markov Decision Processes (MDP) (Kaelbling et al. 1996). The goal is to find an optimal *policy*, that is, a mapping between states and actions, that provides the maximum discounted expected reward $V(s) = E\{\sum_{t=0}^{\infty} \gamma^t r_t\}$ in each state of the space state, where the γ parameter sets the influence of future rewards and r_t is the immediate reward in time t. Different families of RL algorithms solve the problem. In our framework, we will use two temporal difference (TD) algorithms: Sarsa(λ) (Sutton & Barto 1998) and Q-learning (Watkins & Dayan 1992), with an ϵ-greedy exploratory policy. The value of ϵ decays with the number of episodes. One RL process per agent is carried out simultaneously and independently so that each agent perceives the rest of the agents as a part of the environment.

The virtual 3D environment consists on a narrow corridor of 15 m. long and 2 m. wide (Figure 1 on the right). Inside, two groups of embodied agents are initially placed at the ends of the corridor. The goal of each agent is to reach to the opposite end of the corridor. The agents are represented by a cube surrounded by a circle with radius 0.3 m. that represents the collision bounding area. The ODE module performs collision-detection using spheres with the same radius. This module models the collision taking in account friction forces between agents and between an agent and the soil (Martinez-Gil et al. 2012*a*). The state for each agent is described by the features shown in the table of the Figure 1 (left). The agent's sensorization is displayed in Figure 1 (center). The chosen features have been used previously in pedestrian models and they are considered relevant for the kinematic description of the pedestrian (Robin et al. 2009). The agent's actions modify its velocity vector. This variations have also been used to control the trajectories in pedestrian models (Bierlaire & Robin 2009). In an agent's decision, the actions are taken in pairs, which modify the speed (increasing or reducing) and the orientation of the velocity vector (clockwise or counterclockwise). There are eight different ratios plus the 'no operation' option for both the speed and the orientation, resulting in 81 possible combined actions. The maximum number of actions per episode allowed (steps) is 70 but the agents actually use about 30.

In real-valued state spaces, the use of a generalization method is necessary. We compared two methods in our scenario: Vector Quantization (VQ) (Gray 1984) and Tile coding (Sutton & Barto 1998). VQ is a clustering method that uses a finite set of vectors (named codewords) as a codebook to describe the state space. A metric maps each real state with the nearest vector of the codebook (in our case, the euclidean distance is used because of the geometric nature of the features of the state space). The codewords are calculated using the Generalized Lloyd Algorithm clustering method (GLA)(Linde et al. 1980). The data for the clustering are gathered from the sensorizations of the agents inside the virtual environment and each agent builds its own quantizer. The codebook is used as the entry of a tabular value function where the value of each entry in terms of accumulated reward is calculated through the agent's interaction with the environment. The number of codewords has been empirically selected to 8192. Tile coding is a specific case of linear function approximation with binary, sparse features. The value function for each state-action pair is represented as a lineal combination of the parameters as $V_t(s) = \sum_{i=1}^{n} \theta_t(i)\phi_s(i)$ in which the $\phi_s(i)$ features have binary values. Whatever

Sag	Speed of the agent.
Av	Angle of the velocity vector relative to the reference line.
$Dgoal$	Distance to the goal (exit door).
$Srel_i$	Relative speed of the ith nearest agent.
Dag_i	Distance to the ith nearest agent.
Aag_i	Angle of the position of the ith nearest agent relative to the reference line.
Lag_i	Label to identify the group that the neighbor belongs to.
Dob_j	Distance to the jth nearest wall.
Aob_j	Angle of the position of the jth nearest wall relative to the reference line.



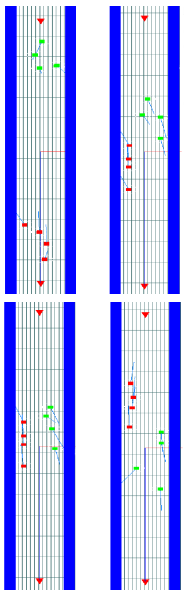


Figure 1: Left: List of the state space features. Center: Agent's reference system for the sensorization. The reference line joins the agent with its goal. Right: Screen shots of four steps of a simulation from the crossing scenario with the ITVQQL schema. The time follows this sequence: top and left, top and right, bottom and left, bottom and right.

the number of dimensions of the space is, it is divided in partitions named tilings. Each element of a specific tiling is a tile and there is only one active tile per tiling; therefore the total number of active binary features is always the same as the number of tilings and was determined empirically to 64 in this domain.

To integrate VQ and Tile coding with a RL Temporal Difference method, the following schemas have been followed:

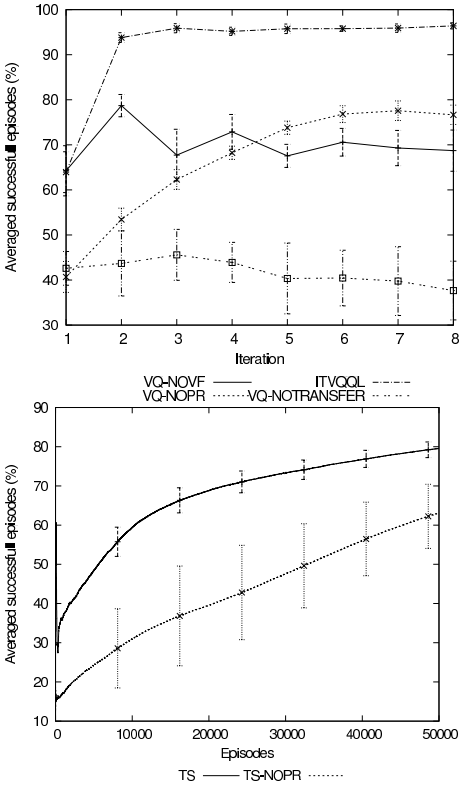
1. Iterative Vector Quantization with Q-Learning (ITVQQL). The VQ generalization method is combined with the Q-learning algorithm in a iterative schema that carries out several learning processes over the same task. The learned value function V_i learned in one learning process l_i that uses the codebook VQ_i is used in a simulation phase to collect policy-biased sensorization data to calculate a new VQ_{i+1} that represents the state space more accurately than VQ_i. Then a new learning process l_{i+1} is carried out using VQ_{i+1} for learning V_{i+1}. Additionally two knowledge transfer methods are used in this schema:

   (a) First, the learned value function of iteration l_i is transferred to initialize the new value function of iteration l_{i+1}. The values of the new value function are loaded with the values of the most similar codeword entry of the old value function. The similarity between codewords of different VQs is given by the euclidean distance. This transfer can be considered a simple case of transfer of an inter-task value function between different representations of the state space, named *complexification* in (Taylor & Stone 2007). This transfer is not a mere initialization because the VQ codebooks of two consecutive iterations are different.

   (b) Second, we have used Probabilistic Policy Reuse (PPR) (Fernández et al. 2010) to incorporate domain knowledge. In PPR, a policy (π_0) is used as a bias in the exploration-exploitation trade-off. Specifically, the policy π_0 is used with a probability ψ that decays exponentially in the number of episodes while the ϵ-greedy exploratory policy is used with probability $1 - \psi$. In our problem, the policy π_0 always suggests the use of an action that drives the agent towards one side of the corridor [1]. It is important to note that PPR is used as a way of exploring efficiently the space of policies to find a solution. If the agent does not find useful to follow the policy π_0 in a state, it will learn a better policy because the exploratory policy is active along all the learning process.

2. Tile coding with Sarsa(λ) (TS). In this approach, each agent carries out a single learning process using Tile Coding and the Sarsa(λ) algorithm. The PPR method described above is also applied in this case.

[1] Specifically, the policy π_0 choose randomly from the set of actions that turns the agent's velocity vector towards the right side of the corridor



Number of agents	8
α	From 0.3 to 0.15
γ	0.9
Initial value of ϵ	1.0
Iterations	8
Episodes per iteration	50000
Reward Goal reached	100
Number of codewords	8192

Number of agents	8
α	0.004
γ	0.9
Initial value of ϵ	1.0
λ	0.9
Episodes	50000
Reward Goal reached	100
Number of tilings	64

Figure 2: Left: Averaged performance for the ITVQQL (top) and the TS (down) schemas. It is measured as the number of times that an agent reach to its goal independently of the rest of agents. In the ITVQQL curves, each point is the average performance of all the agents at the end of each iteration of the learning process. In the TS schema, each curve is the average performance of all the agents along a single learning process. Means are over the 8 agents. Right: The configuration for ITVQQL (top) and TS (down) processes. The parameters have the usual meaning of the RL literature.

3 Experiments and results

Four experiments with the same configuration have been carried out for the ITVQQL approach. The first experiment (titled as ITVQQL in the graphic) uses both, the transfer of the value function and the PPR methods. The second (titled VQ-NOPR) uses the transfer of value function only, the third one (VQ-NOVF in the graphic) uses PPR only, the fourth (titled NO-TRANSFER) does not use any transfer technique. Figure 2 shows the performance (percentage of times that an agent reach to its goal independently of the rest of agents) obtained at the end of each learning process. The ITVQQL curve gets the highest average performance. This iterative schema converges in the third iteration. The curve VQ-NOVF has a similar value than the ITVQQL curve in the first iteration because in that iteration both experiments have the same configuration (there is not value transfer in the first iteration). The same occurs with the VQ-NOPR and the VQ-NOTRANSFER curves. The gap between the curve VQ-NOPR and the curves that use PPR at the first iteration, indicates that the bias provided by the policy π_0 through PPR is very useful for the learning process. For curve VQ-NOPR, the iterative schema is useful from the iteration number 1 to 7. This fact shows that the transfer of the value function needs more iterations when it is used alone than when it is used in combination with PPR. The difference between the VQ-NOTRANSFER curve and the rest of curves reveals the benefits of using transfer techniques. In the VQ-NOTRANSFER experiment, the iterative schema is not useful, likely because the learning process is not long enough to generate better VQs between consecutive iterations. In the second row of the Figure 2 the results of the TS approach are reported. The curve labeled as TS represents data of an experiment that uses PPR while the curve named TS-NOPR does not use PPR. These curves correspond to a single learning process. The initial gap between the two curves is typical in a knowledge transfer process. The final performance is very different in both curves indicating the beneficial effect of the PPR in the TS curve during the learning process.

The performance analysis in simulation is represented in the Table 1. The performance in simulation is different to the average performance shown in Figure 2. The performance in simulation gives the percentage of correct simulations, that is, the simulations in which all the agents reach to the correspondent goal. The first column shows that the performance is similar for the two generalization methods when using knowledge transfer. The second column, that corresponds to the results of the experiments without PPR, shows a decrement of the performance. The percentages are relevant enough to

	WITH TRANSFER	NOPR	NOTV	NOTRANSFER
ITVQQL	81 ± 4	30 ± 4	8 ± 2	0
TS	80 ± 3	68 ± 4	–	–

Table 1: Analysis of the performance in simulation. Mean of the percentage of episodes that end successfully from a series of 100 episodes. In a successful episode, all the agents reach to the correspondent goal. The mean of ten series is displayed.

indicate that the emergent collective phenomena depends on whether the additional information is provided by the PPR transfer method. Additionally, the influence of the use of PPR is higher in the ITVQQL schema than in the TS approach. The value of the third column also indicates that the use of transfer of the value function has a deep influence in the performance of the ITVQQL schema. The value of the NOTRANSFER column shows that the crossing problem is not solved correctly, and in simulation, lane formation can not be clearly observed. In terms of learning, the TS algorithm is more efficient than the ITVQQL. When no transfer techniques are used, the performance of the TS schema is higher (68%) than when the ITVQQL is used (0%). The Figure 1 shows four moments of a simulation for the ITVQQL schema. In both approaches (ITVQQL and TS), the lane formation has similar visual appearance. Videos can be seen in the URL http://www.uv.es/agentes/RL/index.htm.

4 Conclusions

From the results of the experiments described in the previous section we can derive the following conclusions:

- The lane formation is an emergent collective behavior that appears independently of the generalization approach used. However, it requires a learning bias in the exploration process, which is provided through the PPR method.
- The ITVQQL and TS schemas show similar performance results in simulation when the knowledge transfer techniques are active. However, the ITVQQL schema is computationally more expensive than TS because it has to carry out several learning processes.
- The use of knowledge transfer techniques improves the performance of both schemas. Specially, the transfer of value function technique is important in the TRVQQL schema. However, TS is more efficient than ITVQQL when knowledge transfer techniques are not used.

References

Bierlaire, M. & Robin, T. (2009), Pedestrians choices., *in* 'Pedestrian Behavior', Emerald, pp. 1–26.

Fernández, F., García, J. & Veloso., M. (2010), 'Probabilistic policy reuse for inter-task transfer learning.', *Robotics and Autonomous Systems.* **58(7)**, 866–871.

Gray, R. M. (1984), 'Vector quantization', *IEEE ASSP Magazine* **1**(2), 4–29.

Helbing, D., Buzna, L., Johansson, A. & Werner, T. (2005), 'Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions', *Transportation Science* **39**(1), 1–24.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), 'Reinforcement learning: A survey', *Jour. of Artif. Intell. Research* **4**, 237–285.

Linde, Y., Buzo, A. & Gray, R. (1980), 'An algorithm for vector quantizer design', *IEEE Trans. on Commun.* **28**(1), 84–95.

Martinez-Gil, F., Lozano, M. & Fernández, F. (2012*a*), Calibrating a motion model based on reinforcement learning for pedestrian simulation, *in* 'Motion in Games 2012. LNCS 7660. M. Kallmann, K. Bekris (Eds.)', Springer.

Martinez-Gil, F., Lozano, M. & Fernández, F. (2012*b*), Multi-agent reinforcement learning for simulating pedestrian navigation, *in* 'Adaptive and Learning Agents (ALA'11). LNCS, vol 7113', Springer, pp. 54–69.

Robin, T., Antonioni, G., Bierlaire, M. & Cruz, J. (2009), 'Specification, estimation and validation of a pedestrian walking behavior model', *Transportation Research* **43**, 36–56.

Schödl, A. & Essa, I. (2000), Machine learning for video-based rendering, *in* 'Advances in Neural Information Processing Systems', MIT Press, pp. 1002–1008.

Sutton, R. S. & Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

Taylor, M. & Stone, P. (2007), Representation transfer in reinforcement learning, *in* 'AAAI 2007 Fall Symposium on Computational Approacher to Representation Change during Learning and Development'.

Treuille, A., Lee, Y. & Popović, Z. (2007), 'Near-optimal character animation with continuous control', *ACM Transactions on Graphics (SIGGRAPH 2007)* **26**(3).

Watkins, C. & Dayan, P. (1992), 'Q-learning', *Machine Learning* **8**, 279–292.

Human reinforcement learning processes act on learned attentionally-filtered representations of the world

Yuan Chang Leong
Department of Psychology
Princeton Neuroscience Institute
Princeton University
Princeton, NJ 08544
yl@princeton.edu

Yael Niv
Department of Psychology
Princeton Neuroscience Institute
Princeton University
Princeton, NJ 08544
yael@princeton.edu

Abstract

Reinforcement learning (RL) models are often applied to study human learning and decision-making. However, simple RL algorithms do not fare well in explaining learning behavior in real world situations where the environment is high-dimensional and the relevant states are not known. As a solution, we propose that RL processes act on an attentionally-filtered representation of the environment. This improves the computational efficiency of RL by constraining the state-space that the learning agent has to consider. We further propose that the attention filter is learned and is dynamically modulated according to the outcomes of ongoing decisions. To test our hypotheses, we had participants perform a decision-making task with multi-dimensional stimuli and probabilistic awards. Model-based analysis of participants' choices suggests that participants prefer strategies that favor computational efficiency at the expense of statistical optimality. To better study the dynamics of attention, we had a group of participants perform a variant of the task in which they had to select the dimensions they wanted to view before making their choice. We treated the viewed dimensions as a proxy for participants' attention filter. Our models fit the data better when learning was restricted to attended dimensions, suggesting that participants do indeed constrain choice and learning to a subset of dimensions. Finally, attention dynamics themselves were best explained by a model that preferentially attended to dimensions with features that have acquired high value over the course of learning. This result provides evidence that the attention filter is dynamically modulated as participants receive feedback from ongoing decisions.

Keywords: attention, state representation, function approximation, active-sensing, human decision-making

Acknowledgements

We are grateful to Angela Radulescu, Reka Daniel and Andra Geana for their invaluable input to this project. This work was supported in part by NIH grant R01MH098861 and by an Alfred P. Sloan Research Fellowship and a New Scholar award from the Ellison Medical Foundation to YN.

1 Introduction

The framework of reinforcement learning (RL) has had a tremendous impact on the fields of psychology and neuroscience. In particular, the temporal difference (TD) learning model has helped advance our understanding of animal and human learning by providing a mathematically precise definition of how an agent learns the association between predictive stimuli and rewards [1]. In laboratory controlled experiments, where the state-space is well-defined, TD learning has indeed provided a good account for both behavioral and neural data [2, 3]. Real-world learning, however, takes place in a highly complex and multidimensional environment, which poses a challenge to the TD learning framework.

In particular, it is a well-known problem in operations research and machine learning that the number of states of a task increases exponentially with increasing number of dimensions on which these states are defined. This is known as the *curse of dimensionality* [4]. Since TD learning assigns values to states (or state-action pairs), the amount of experience required to arrive at an approximately correct value for all states increases with the number of states. Yet both animals and humans can solve complicated learning problems with limited experience.

We propose that efficient learning is possible because people employ selective attention as a learning strategy [see also 5, 6]. The role of selective attention in regulating cognitive processes is well established [7]. Here, we hypothesize that attention facilitates learning by carving out the state-space that RL operates on. We further postulate that this attention filter is learned, and as such, is dynamically modulated by the outcomes of ongoing decisions.

To test our hypotheses, we had human participants perform 3-armed bandit tasks with multidimensional stimuli and probabilistic rewards. We found that participants opt for computationally efficient strategies at the expense of statistical optimality. In addition, we found that participants' attentional-selection strategy was best described by a model that allocates attention according to the learned values of stimuli. As the values are updated with ongoing learning, the attentional filter is also modulated. The current results support our hypothesis that learning is constrained by attention, but we also learn what to attend to.

2 Experimental Tasks

2.1 Faces/Houses/Tools (FHT) Task

The FHT task was designed to be a simplified analog of many real world problems, where people have to make decisions based on multidimensional information under conditions in which most dimensions are uninformative to the decision at hand. On each trial, participants chose between three bandits, each described by features from three different dimensions: a face, a house and a tool (Figure 1a). Stimuli on each trial were generated by randomly recombining features from each dimension. In any one 'game' only one dimension (e.g., tools) was relevant to determining reward and only one target feature in that dimension (e.g., saw) was associated with a high reward probability ($p = 0.75$). Choosing stimuli that did not contain this feature yielded a reward with only $p = 0.25$. Participants were not told which dimension was relevant, and were tasked with getting as much reward as possible. Eighteen participants performed the FHT task and were paid \$12-\$15 according to their performance. Each participant played 56 games of 25 trials each.

2.2 Active Sensing Faces/Houses/Tools (asFHT) Task

To study attention processes in the FHT task, we had a separate group of participants play a variant of the task where they had to select, via button presses, the dimensions they would like to view before making their choices. On each trial, they could choose to view as many dimensions as they wish, before selecting a stimulus (Figure 1b). Recent work suggests that attentional sampling is an active process that shares many similarities with sensorimotor sampling routines [8]. As such, it is reasonable to assume that the dimensions participants chose to view would also be the dimensions they would have chosen to attend to. Nineteen participants performed the asFHT task and were paid \$12-\$15 according to their performance. The experiment began with 10 games of the FHT task to allow participants to familiarize themselves with the task structure. Participants then played 25 games (25 trials each) of the asFHT task.

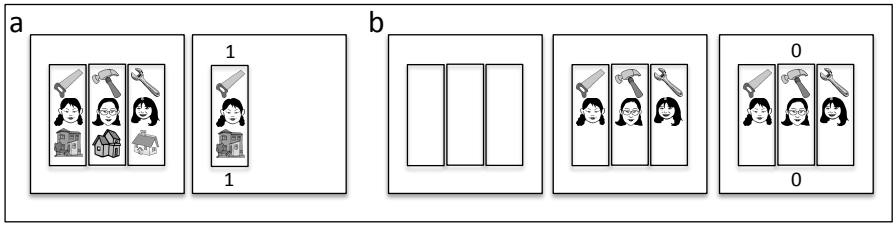


Figure 1: Stimulus displays for *a.* FHT Task and *b.* asFHT task.

3 Model-Based Analysis

3.1 Models of Choice Behavior

We analyzed participants' choice behavior in the FHT task using three computational models[1]. The first model is a *Bayesian model* that uses Bayes' rule to infer the posterior distribution that each feature f in dimension d is the target feature given all previous data $D_{1:t}$ and rewards $r_{1:t}$

$$p(d, f|D_{1:t}, r_{1:t}) \propto p(r_t|d, f)p(d, f|D_{1:t-1}, r_{1:t-1}). \tag{1}$$

This distribution is then used to compute the probability of reward for each stimulus S_i [5, 6]:

$$V_t(S_i) = p(r_t = 1|S_i, D_{1:t-1}) = \sum_d p(d, f \in S_i|D_{1:t-1}, r_{1:t-1})\rho_h + (1 - p(d, f|D_{1:t-1}, r_{1:t-1})\rho_l \tag{2}$$

where ρ_h is the probability of reward for the target feature and ρ_l is the probability of reward for a non-target feature. This model incorporates all available information across all dimensions and features in an statistically accurate manner. As such, it assumes statistically optimal learning about all features at once, akin to a diffused focus of attention.

The second model is a *function approximation* (FA) model that learns a weight w for each of the nine features. It then computes the value of stimulus S_i as the average of feature weights of this stimulus:

$$V_t(S_i) = \frac{1}{n}\sum_{d=1}^{n} w_t(d, f \in S_i) \tag{3}$$

The feature weights for the chosen stimulus c_t are updated according to TD learning [1]:

$$\delta_t = r_t - V_t(c_t) \tag{4}$$

$$w_{t+1}(d, f \in c_t) = w_t(d, f \in c_t) + \frac{1}{n}\eta\delta_t \qquad \forall d = 1, 2, 3 \tag{5}$$

where δ_t is the prediction error for trial t, n is the number of dimensions and η is the learning rate. The FA model is less statistically optimal than the Bayesian model as it learns point estimates of feature weights and does not maintain the full posterior distribution over all features. It also learns only about chosen features, and thus assumes a stronger focus of attention than the Bayesian model.

Finally, the *Decay model*, is identical to the FA model, but in addition, in this model weights of unchosen features decay to zero:

$$w_{t+1}(d, f \notin c_t) = (1 - \eta_k)w_t(d, f \notin c_t) \qquad \forall d = 1, 2, 3 \tag{6}$$

where η_k is the decay rate. The Decay model is the least statistically optimal model of the three as it loses information about unchosen features on each trial. However, it assumes the strongest attention focus since only features that have been consistently attended to and chosen can acquire weights significant enough to influence choice.

3.2 Models of Attention

In general, there are seven possible combinations of viewed dimensions on each trial: one dimension only (three possible options), a combination of any two dimensions (three possible combinations) or all three dimensions.

To model the dimensions participants chose to view on each trial, in the *Dimension Value* model we assumed that attention is costly. As such, participants must balance the benefits of attending to a dimension with the associated cost. The model thus computes the value of each attention combination a_t as the utility of attending to the dimensions in that combination minus a cost that scales with the number of attended dimensions:

$$V_t(a_t) = \sum_{d \in a_t} U_t(d) - nJ \tag{7}$$

where $U(d)$, the utility of attending to dimension d, was determined by summing the weights of features along d, such that the utility of viewing a dimension would be higher when the model has learned high weights for features in that dimension. In equation (7), n is the number of dimensions attended to and J is a cost penalty per dimension attended.

As a baseline for comparison with this model, we tested a model in which participants narrow their focus of attention over time regardless of the specific utility of each dimension. In this *Game Horizon* model, the utility of attending to each dimension is fixed (i.e., $U_t(d) = K$), but the cost of attending to dimensions increases over the course of the game:

$$V_t(a_t) = nK - nJT \tag{8}$$

where n is the number of dimensions viewed in option a_t, and T is the trial number in the current game.

[1]Choice behavior in the asFHT task was analyzed using the same three models. However, the models were modified such that both value computation and update depended only on viewed dimensions.

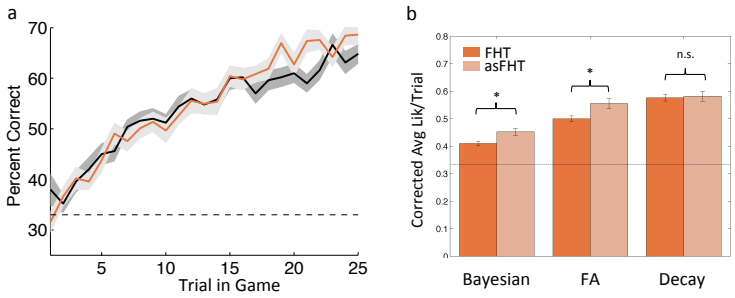


Figure 2: Participants' choice behavior: *a.* Learning curves for the FHT task (black) and asFHT task (orange). *b.* Model performance on data from FHT Task (dark shading) and asFHT Task (light shading). Models were evaluated using corrected average likelihood per trial. Error bars denote SEM; dashed lines denote chance level (33%); * denotes $p < 0.05$.

3.3 Model Comparison and Parameter Estimation

For all models, a "softmax" policy was used to compute the probability of making a particular choice[2], $\pi(c)$, or allocation of attention, $\pi(a)$, on each trial:

$$\pi_t(x) = \frac{e^{\beta V_t(x)}}{\sum_i e^{\beta V_t(i)}} \tag{9}$$

where $x = c$ or a, i enumerates all actions available to the model on that trial, and β is an inverse temperature parameter that determines the balance between exploration and exploitation.

Model parameters were optimized by finding participant-specific parameters that maximized the log likelihood of the participant's data given the model. These parameters were then used to compute the Bayesian Information Criterion (BIC) approximation of model evidence [9], E_M:

$$E_M \approx log(p(D|M, \hat{\theta}_M)) - \frac{||\hat{\theta}||}{2} log\ N \tag{10}$$

where $p(D|M, \hat{\theta}_M)$ is the likelihood of data D given model M and parameters $\hat{\theta}_M$, $||\hat{\theta}||$ is the number of free parameters in the model and N is the number of data points (trials). To provide a more intuitive measure of model evidence, we divided the total score for each model by the number of trials for which a participant provided a response, and exponentiated it, to yield a complexity-corrected average likelihood per trial that varies between 0 and 1.

4 Results

4.1 Choice Behavior

We first evaluated participants' performance by calculating the percentage of trials on which participants chose the stimulus containing the target feature. Figure 2a shows performance as a function of trial within a game. A between-subjects repeated measures ANOVA did not find a main effect of task type ($F(24, 1) = 0.32, p = 0.57$), indicating that there was no significant difference between participants' learning of the two tasks.

As is clear from Figure 2b, all models of choice behavior performed considerably better than chance (one-tailed t-tests, $p < 0.001$). For both tasks, the Decay model was best supported by the data (two-tailed t-tests, $p < 0.001$). The Bayesian and FA models fit data from the asFHT task better than that from the FHT task (two-tailed t-tests, $p < 0.05$). There was no significant difference between the fits of the Decay model for the two tasks ($t(35) = 0.72, p = 0.24$).

4.2 Attention

In the asFHT Task, the number of viewed dimensions decreased over the course of each game (trial 1: Mean = 2.34, SE = 0.17; trial 25: Mean = 1.67 , SE = 0.11; Figure 3a). Both the Game Horizon model and the Dimension Value model predicted participants' focus of attention better than chance (one-tailed t-tests, $p < 0.001$). However, the Dimension Value model performed significantly better than the Game Horizon model ($t(18) = 9.1, p < 0.001$, Figure 3b).

[2]For the attention models, feature weights were first generated using the best-fitting model of choice behavior.

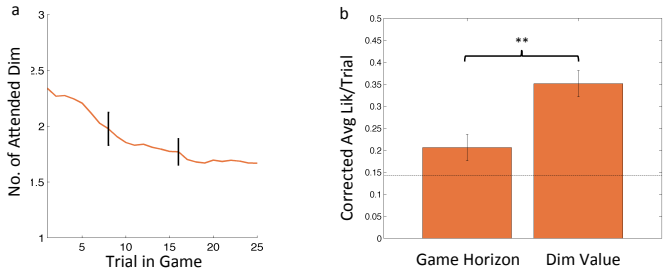


Figure 3: Dynamics of attention processes in the asFHT Task. *a.*. Average number of attended dimensions over the course of a game. *b.* Model performance on predicting participants' focus of attention. Error bars indicate SEM; dashed line indicates chance level (0.14%); ** indicates $p < 0.001$.

5 Discussion

Our experimental tasks were designed to mimic some aspects of the cluttered multidimensional state in which real-world learning and decision making are embedded in. While participants could learn to solve these tasks, an analysis of their choice behavior revealed that they were not doing so in a statistically optimal manner. Consistent with previous work [5, 6], we found that participants' choice behavior was best explained by strategies that were computationally efficient, but statistically suboptimal. In particular, the best-fit "Decay model" was an RL model with a function approximation architecture that also decays weights of unchosen features. This model learns only about chosen features, and loses useful information on each trial. However, it places low demands on computational resources by relying on algorithms that are less computationally costly and thus can easily scale up to many dimensions. Such a strategy may reflect a necessary compromise between optimal learning and computational demands given limited resources.

In some senses, the Decay model also assumes the narrowest focus of attention. We had hypothesized that attention might serve to further reduce computational demands by carving out a suitable state-space for efficient learning. The asFHT task was designed to directly test this hypothesis. Interestingly, despite the fact that participants limited their viewing to only a subset of dimensions, comparison of learning curves indicated no difference in learning between this and the FHT task in which all dimensions were always available. Moreover, incorporating information about participants focus of attention significantly improved the model fits for both that Bayesian model and the FA model. That is, even though participants in the FHT task were presented with all dimensions, they might have been learning and making choices based only on the subset of dimensions that they were attending to. Finally, it is noteworthy that performance of the Decay model was not significantly different between the two tasks. We suggest that this is because the Decay model implicitly implements a selective attention component for both tasks: due to the weight decay, weights of features that are not consistently chosen (presumably because they are not being attended to) decay to zero and thus choice in this model comes to be driven by features that have been consistently attended to. This interpretation is compatible with a role for explicit attentional mechanisms in learning and decision-making, though further work needs to be conducted to formally explore the relationship between the Decay model and attention.

Lastly, we were interested in understanding how the focus of attention changes with learning. Here we found that participants' focus of attention depended on the value of attending to the different dimensions. In the beginning of each game, when participants had no information about any of the dimensions, it was worthwhile to attend to multiple dimensions. However, as they learned more about each dimension, it was more efficient to attend only to dimensions with high-value features. Taken together, these results demonstrate an intricate relationship between learning and attention—attention constrains what we learn about, but we also learn what to attend to.

References

[1] Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction. Cambridge*, MA: MIT Press.
[2] Niv, Y. (2009). *Reinforcement learning in the brain. Journal of Mathematical Psychology*, 53(3), 139154.
[3] Balleine, B. W., Daw, N. D., & O'Doherty, J. (2009). Multiple forms of value learning and the function of dopamine. In P. W. Glimcher, C. F. Camerer, C. Fehr, & R. A. Poldrack (Eds.), *Neuroeconomics: Decision Making and the Brain* (pp. 367387). London: Academic Press.
[4] Bellman, R. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press
[5] Gershman, S.J., Cohen, J.D., and Niv, Y. (2010). Learning to selectively attend. In *32nd Annual Conference of the Cognitive Science Society*, Portland
[6] Wilson, R. C., & Niv, Y. (2011). Inferring relevance in a changing world. *Frontiers in human neuroscience*, 5, 189.
[7] Miller, E K, & Cohen, J. D. (2001). An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24, 167202.
[8] Schroeder, C. E. et al. (2010). Dynamics of Active Sensing and perceptual selection. *Current opinion in neurobiology*, 20(2), 1726.
[9] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 461- 464.

Modeling Experiential Knowledge: Limitations in Learning Non-Linear Dynamics for Sustainable Renewable Resource Management

Emilie A.L. Lindkvist*
Stockholm Resilience Center
Stockholm University
106 20 Stockholm
Sweden
emilie.lindkvist@stockholmresilience.su.se

Jon Norberg
Stockholm Resilience Center
Stockholm University
106 20 Stockholm
Sweden
jon.norberg@stockholmresilience.su.se

Abstract

Adaptive management of renewable natural resources seek to include the knowledge and experience of resource users by incorporating learning by doing (LBD) in the management process to ensure sustainable resource use, in presence of uncertainty and environmental change. By contrast, an optimal management approach identifies the most efficient exploitation strategy by postulating an a priori understanding of the resource dynamics, and assumes an analytical solution can be formulated. A particular wicked problem in resource management is threshold dynamics, where the effect of passing a threshold switches the feedbacks within the system and changes the provisioning rate. Recovery is then constrained by the degree of lock-in (s.c. hysteresis). Here we study the limitations and possibilities of LBD in achieving optimal management by using the analytical solution of a generic resource growth function as a benchmark for evaluating the performance of an agent equipped with state of the art learning features. The agent uses Reinforcement learning (SARSA), a radial basis function network, and Softmax decision-making. We study four learning parameters; learning rate of mental model, eligibility trace decay rate, discount rate, and the level of exploration. We let the agent solve a logistic growth function (e.g. a fish stock) and compare the results of LBD when managing this resource with or without a threshold effect. We show that for a logistic growth function a LDB agent can sustainably manage the resource with 90% efficiency compared to optimal control, whereas when we add a threshold behavior to the resource function this efficiency drops to 65%. To achieve the highest possible outcome the following features are of outmost importance; an adequate degree of experimentation, high valuation of future stocks (discounting), and a modest learning rate. We finally conclude that for these problems learning through hindsight (eligibility trace) has limitations.

Keywords: Natural Resource Management; Renewable resources; Non-Linear Systems; Hysteresis; Adaptive Management; Learning by Doing; Radial Basis Function Networks

Acknowledgements

Stockholm Resilience Center for financial support, and Örjan Ekeberg (Royal Institute of Technology, Stockholm) for help on the methodological design.

*http://www.stockholmresilience.su.se/

1 Introduction

Global environmental drivers are changing as a consequence of human activities [4]. The direction and magnitude of these changes varies locally, and our capacity to understand and predict the implications of these changes is limited. This limitation is particularly manifested when local impacts of global change are transferred down through complex ecosystems. Thus, local knowledge is not only crucial but needs to continuously adapt to both anticipated change and sudden fluctuations. Within sustainability sciences and natural resource management, adaptive management has been championed as an advantageous approach for dealing with these unexpected dynamics, by incorporating local knowledge and learning-by-doing (LBD) into the management process[1]. By contrast, optimal management identifies the most efficient exploitation strategy by postulating an a priori understanding of the resource's dynamics, and assumes an analytical solution can be formulated. Further, a particular wicked problem in renewable resource management is threshold dynamics, where the effect of passing a threshold switches the feedbacks within the system and changes the provisioning rate. Recovery is then constrained by the degree of lock-in (s.c. hysteresis). To ensure future sustainable use of natural resources, increased understanding of the implications of adaptive management, and its corresponding learning process in relation to the resources's feedbacks, is crucial.

In this paper we compare the two management processes through a stylized resource management problem, using a fish population with logistic growth that may or may not exhibit hysteresis, and let Reinforcement learning (SARSA) represent the LBD process, and a radial basis function network [8] represent the local knowledge of an agent. The agent, here representing a fisherman or a cooperating unit of fishermen, have no a priori understanding of the resource's dynamics and learn solely by resource interaction. Our interests are 1) How does the LBD process respond to different levels of resource complexity? 2) How are classic natural resource management dilemmas affected by the trade-offs between exploration and exploitation (Softmax), the level of discounting, learning from past experiences (eligibility trace), and fast vs. slow mental model learning rate, depending on resource's complexity?

Similar work within resource management is conspicuously lacking and we aim to provide useful insights to the discourse on the role of learning in natural resource management, and to highlight trade-offs in classic natural resource management dilemmas through this novel approach.

2 The Model

We created a setting with a model of a resource system and a computer agent, where the agent has the ability to interact with the resource, to process and store experiences, and to make decisions based on these experiences. Each time step (fishing event), the agent could set its harvest effort (action) and observe the yield. We let the agent interact with two resources, both of which were represented by hypothetical single species regenerative populations, but had different internal dynamics [3]. The internal dynamics included logistic growth rate but with or without a threshold effect, and will further on be referred to as the *logistic function* and the *threshold function*, to state which dynamics are in focus.

2.1 Resource Dynamics and Agent's Maximization Problem

The goal of the agent was to find the effort resulting in the maximum economic yield (MEY) over time. The economic yield was calculated as the value of harvest, minus the cost of harvesting;

$$r_t = pa_t s_t - ca_t \tag{1}$$

where r is the reward, p the price of fish, a the effort, s the biomass, and c the cost of fishing. In our simulations we let $p = 1.0$, and $c = 0.1$ thus a higher effort generated linearly a higher cost of fishing. The resource system was represented by two functions (i.e. the two levels of complexity);

Logistic Function:

$$\frac{ds}{dt} = gs\left(1 - \frac{s}{K}\right) - as - \epsilon s \tag{2}$$

Threshold Function:

$$\frac{ds}{dt} = gs\left(1 - \frac{s}{K}\right) - q\frac{s^2}{h^2 + s^2} - as - \epsilon s \tag{3}$$

where g is the growth rate, K the carrying capacity, s the biomass, a the effort, q the maximum predation rate, and h sets the level where predation is 50% of the maximum predation, see [3]. Parameter ϵ corresponds stochastic stock removal, mimicking exogenous events (there was a 5% chance that 5% to 95% of the stock was removed). The change in growth over time is shown in fig. 1 A. MEY is found by using the following optimal control function $max(0, 1 + g(1 - s/K) - (K + c)/2s) - q * s/(h^2 + s^2)$. Note that $q = 0$ for the logistic function.

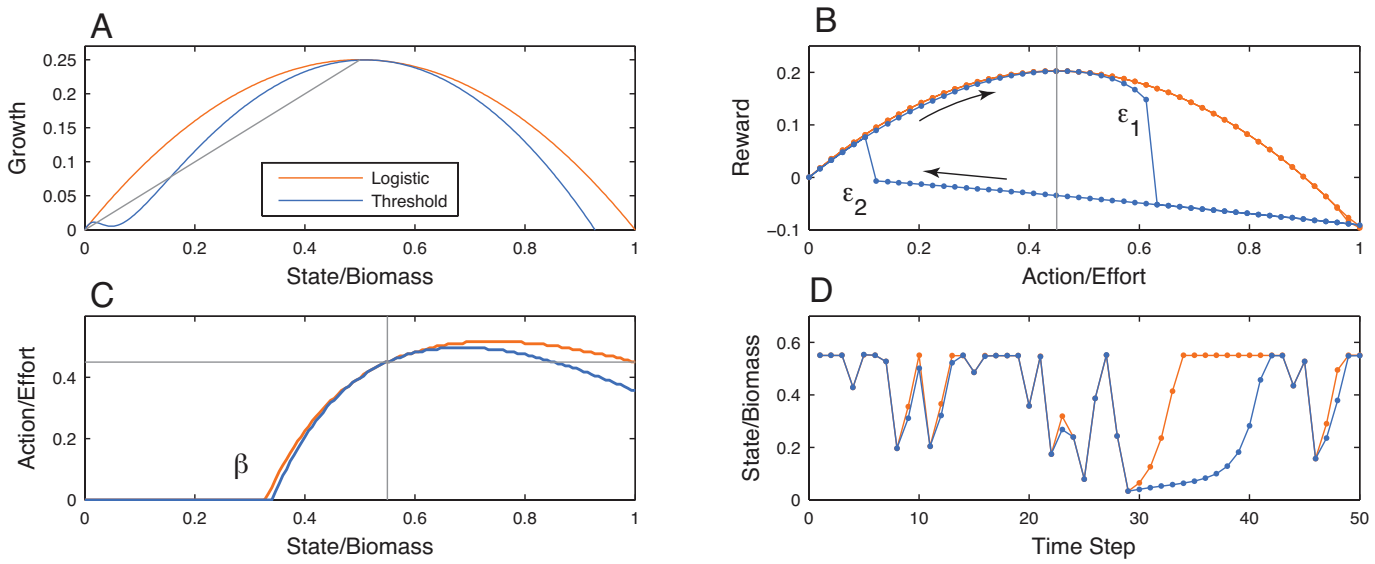


Figure 1: (**A**) Shows the regrowth depending on the current state. (**B**) When applying an action a (a particular value on the x-axis) until a steady state equilibrium is reached (y-axis), the threshold effect takes place for action $> \varepsilon_1$ (threshold function). If the tipping point ε_1 has been trespassed, repeated actions below ε_2 are needed to be able to reach MEY again. (**C**) Shows the optimal control path of the two resource functions, i.e. when the state is below β the optimal action is to stop fishing until the stock is above β. (**D**) effect on state regrowth when enforced mortality has occurred and the action is choosen according to the optimal control.

In fig. 1 B we show the steady state solution in yield (reward, r) for any given effort. While the logistic function has one unique solution for the span of effort $[0, 1]$ the threshold function displays two attractors. Should one increase effort above the threshold ϵ_1, hysteresis is defined as the amount of effort one has to reduce from ϵ_1 to ϵ_2 in order to return to the upper and higher yield attractor. To achieve MEY the state must be kept at $(K + c)/2$. Since the reward is a simple linear function (eq. 1), a most rapid approach solution is valid for both resource functions [3]. This means that the action leading to MEY has to be chosen in order to most rapidly bring the stock towards $(K + c)/2$, as depicted in fig. 1 C. Note that despite the additional threshold term in eq. 3 (generating the hysteresis in fig. 1 B) the the optimal paths of actions are very similar, fig. 1 C.

2.2 The Agent

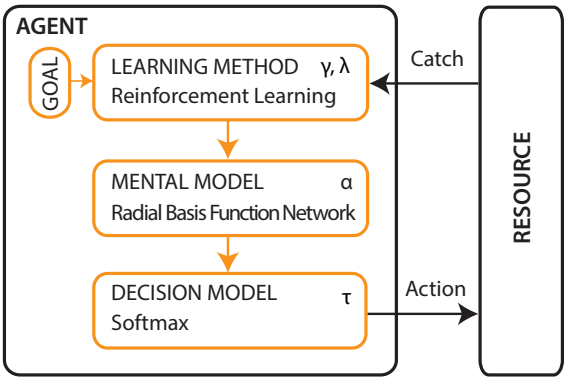


Figure 2: Conceptual model of the agent-resource system. The learning method updated the mental model, and the mental model effected the decision on what action was chosen. Each part used the parameters update rate (α), discount factor (γ), level of hindsight (λ), exploration level (τ), to influence the learning process. The goal is to achieve MEY.

The agent consists of RL (a version of SARSA) for learning, a RBF network for its mental model, and Softmax for decision-making, see fig. 2 . Since the aim is for the setting to represent a real world problem, the concept of epochs ("starting over") is left out of SARSA. The action taken by the agent is discretisized in the Softmax model, to allow for finding the agents perceived best action in relation to a current state. The mental model consists of a number of evenly spread RBFs. Too few RBFs gives a narrow mind lacking the ability to differentiate between different actions and states, while too many RBFs hinders generalization. In our model we used 9^2 RBFs to enable the agent to have the ability to approximate the discontinuity (β in fig. 1 C).

Prediction of Future States: In order to calculate the next prediction ($Q_t(s_{t+1}, a_{t+1})$), an estimate of the future resource state is required. In our model the agent (fisherman) can obviously not predict the future fish stock, instead we chose a simple regression model based on past the fishers past experiences. Letting a represent the action and s the state and μ a regression constant, then the prediction of the next state was calculated as $s_{t+1} = \mu_1 s_t + \mu_2 a_t s_t$. This allowed the agent to understand 1) that the resource system was self generating and 2) that the resource was affected by harvesting.

2.3 Computer Experiments

In our simulations each time step corresponded to a specific discrete time frame, for example a week. During one time step 1) the agent made a decision 2) performed an action 3) received a response and 4) updated its mental model of the resource dynamics (i.e. function approximation). Simulations and analysis were performed using MATLAB 2012. The evaluation criteria for analyzing the performance of the agents was calculated by a performance index, as;

$$PI = \frac{\sum_{t=1}^{T} r_t}{\sum_{t=1}^{T} r^{\star}{}_t} \tag{4}$$

Where t is the time step, r is the agent's reward, and $r^{\star}$ is the reward given by the analytical solution, $T = 300$. The optimal parameter values were found by first running 3000 random combinations of update rate (the learning rate of RBF-network α), discount factor (λ), level of hindsight (eligibility trace's γ), and exploration level (temperature of Softmax τ). Then optimising the best parameter value combinations using a gradient ascent method.

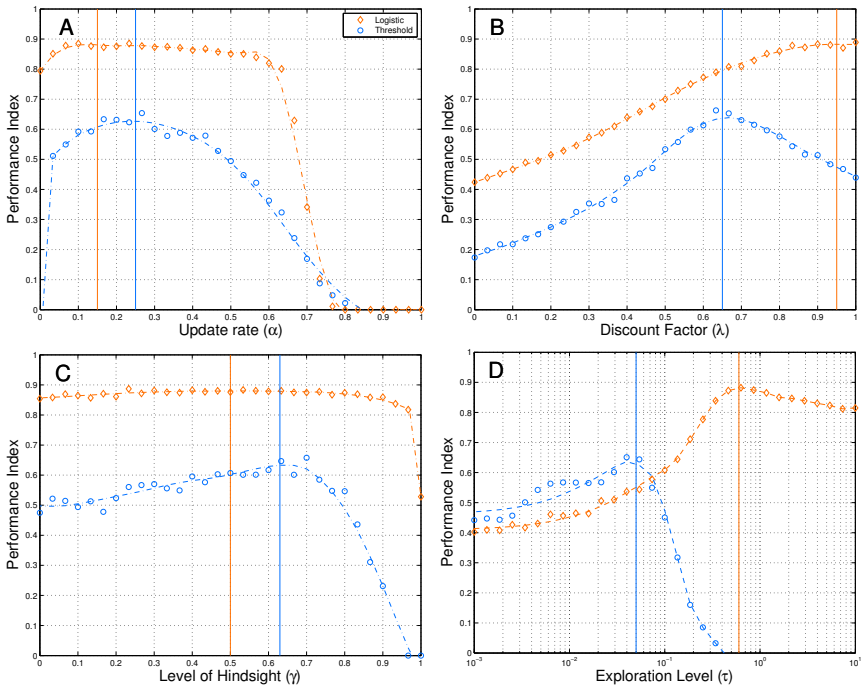


Figure 3: The figure shows the effect on performance (net income) when varying each parameter, while other parameters are held fixed. Negative values are left out from the plot. The fixed values for the logistic function were $\alpha = 0.15$, $\lambda = 0.95$, $\gamma = 0.5$, $\tau = 0.6$, and for the threshold function $\alpha = 0.25$, $\lambda = 0.65$, $\gamma = 0.63$, $\tau = 0.05$. Each dot represents an average of 150 runs while the dash-dot line is a smooth function on the data.

3 Results

Although the learning process is costly, a behavior resulting in 90% (logistic function) or 65% (threshold function) efficiency of the theoretical optimum can be learned (fig. 3). In relation to the four learning parameters we studied, the agent's behavior in solving the logistic function can be described as highly explorative with less increase in reward through the learning process. On the other hand, the agent's behavior when using a resource with a threshold present showed that careful exploration in combination with the learning process is crucial. The importance of the learning parameters for the threshold function was emphasized through the sensitivity to deviations from its optimal parameter values, showing a more rapid drop in performance as one moved away from each fixed optimal parameter value (3 A-D).

4 Conclusion

Considering the complexity of natural resource dynamics, optimal strategies are often impossible to identify [11, 2067], and thus local ecological knowledge can clearly contribute to more sustainable management, if it is part of an adaptive management program as [10] concludes. On the other hand, if these localized experiments in LBD are performed without a broader context and possibly at larger scales, the social-ecological consequences could be devastating given that the system dynamics often are uncertain, unknown or unknowable [5].

Further, a higher negative impact on the performance of resources with threshold dynamics were found when deviating from the fixed optimal parameter values vis-à-vis resources lacking thresholds. This implies that resources with threshold effects require careful attention to management decisions and entail a substantial body of knowledge to avoid mismanagement. This study confirms, in accordance with the results of [2], the inherent problems with applying trial and error methods in a resource management situation where thresholds are present, due to the cost of trespassing a threshold and the difficulty to learn and adapt to the rapid changes in feedbacks. Finally, we conclude that—inadequate consideration of future stocks (discounting), the speed of adapting ones mental model, and that the timeline for hindsight must not be excessive—are particularly important for reaching sustainable outcomes.

This paper [7] introduces a model for studying renewable resource management problems. In Lindkvist and Norberg [6, In prep.] we explore the effects of climate change by continuous and abrupt changes in the growth rate of a fish stock, as these changes are shown to have alarming consequences for our future earth. Hence, by using RL within resource management problems we highlight trade-offs between important learning parameters and their impacts on different levels of resource complexity as well as dynamically changing feed backs, while contributing to the discourse within sustainability sciences, natural resource management and ecological economics on adaptive management and learning.

References

[1] Armitage, D., Marschke, M., and Plummer, R., 2008. Adaptive co-management and the paradox of learning, Global Environmental Change, 18, 86-98.

[2] Biggs, R. and Carpenter, S. R. and Brock, W. A., 2009. Turning back from the brink: Detecting an impending regime shift in time to avert it. Proceedings of the National academy of Sciences, 106, 826-831.

[3] Clark, C. W, 1976. Mathematical bioeconomics: the optimal management of renewable resources, New York.

[4] IPCC, 2007. Climate change 2007: synthesis report, Geneva.

[5] Levin, S. A, 2003. Complex adaptive systems: exploring the known, the unknown and the unknowable. Bulletin-American Mathematical Society, 40, 3-20.

[6] Lindkvist, E., and Norberg, J., In Preparation. Modeling Learning-By-Doing: Managing a Renewable Resource Undergoing Effects of Climate Change.

[7] Lindkvist, E., and Norberg, J., Accepted. Modeling Experiential Knowledge: Limitations in Learning Non-Linear Dynamics for Sustainable Renewable Resource Management. Ecological Economics.

[8] Poggio, T. and Girosi, F.,1989. A theory of networks for approximation and learning. MIT.

[9] Sutton, Richard S. and Barto, Andrew G., 1998. Reinforcement Learning: An Introduction. isbn 0585024456.

[10] Walters, Carl J, 2007. Is Adaptive Management Helping to Solve Fisheries Problems? AMBIO: A Journal of the Human Environment, 36, 304-307.

[11] Walters, Carl J. and Holling, C. S., 1990. Large-Scale Management Experiments and Learning by Doing. Ecology, 71, 2060-2068.

Using subgoals to reduce the descriptive complexity of probabilistic inference and control programs

Domenico Maisto
Institute for High Performance Computing and Networking
National Research Council
Via Pietro Castellino 111, 80131 Napoli, Italy
domenico.maisto@icar.cnr.it

Francesco Donnarumma
Institute of Cognitive Sciences and Technologies
National Research Council
Via S. Martino della Battaglia, 44, 00185 Rome, Italy
francesco.donnarumma@istc.cnr.it

Giovanni Pezzulo
Institute of Cognitive Sciences and Technologies
National Research Council
Via S. Martino della Battaglia, 44, 00185 Rome, Italy
giovanni.pezzulo@istc.cnr.it

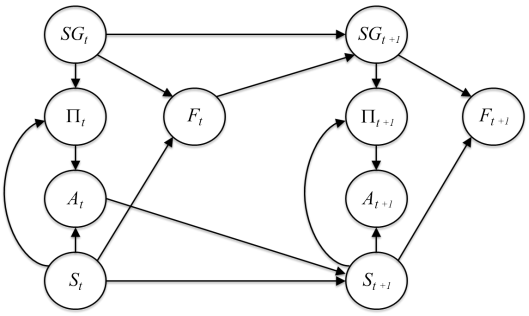
Abstract

Humans and other animals are able to flexibly select among internally generated goals and form plans to achieve them. Still, the neuronal and computational principles governing these abilities are incompletely known. In computational neuroscience, goal-directed decision-making has been linked to model-based methods of reinforcement learning, which use a model of the task to predict the outcome of possible courses of actions, and can select flexibly among them. In principle, this method permits planning optimal action sequences. However, model-based computations are prohibitive for large state spaces and several methods to simplify them have been proposed. In hierarchical reinforcement learning, temporal abstractions methods such as the Options framework permit splitting the search space by learning reusable macro-actions that achieve subgoals. In this article we offer a normative perspective on the role of subgoals and temporal abstractions in model-based computations. We hypothesize that the main role of subgoals is reducing the complexity of learning, inference, and control tasks by guiding the selection of more compact control programs. To explore this idea, we adopt a Bayesian formulation of model-based search: planning-as-inference. In the proposed method, subgoals and associated policies are selected via probabilistic inference using principles of descriptive complexity. We present preliminary results that show the suitability of the proposed method and discuss the links with brain circuits for goal and subgoal processing in prefrontal cortex.

Keywords: goal-directed decision-making; goal; subgoal; temporal abstraction; Bayesian inference; planning-as-inference; descriptive complexity; information compression

Acknowledgements

Research funded by the EU's FP7 under grant agreement no FP7-ICT-270108 (Goal-Leaders).



SG_t	subgoal	$[0, ..., n]$
Π	policy	$[0, ..., m]$
A_t	action	$\{u, d, l, r, \varepsilon\}$
F_t	termination condition	$\{0, 1, 2\}$
S_t	state	$[0, ..., n]$

Table 1: Left: Graphical model (Dynamic Bayesian Network [13]). Right: Stochastic variables

1 Introduction

A widespread idea in neuroscience is that goals and subgoals maintained in prefrontal hierarchies have a key role in exerting cognitive control over behavior [1, 2]. Goals and subgoals are usually assigned several roles: serving as reference states for action selection and monitoring, stabilizing behavior by preventing oscillations between incompatible action patterns, facilitating planning, influencing perception, attention, memory retrieval, and behavior in a top-down way. However, the neuronal and computational mechanisms supporting goal and subgoal processing remain elusive.

In particular, it is only recently that model-based computations and planning have been studied experimentally. A series of monkey studies revealed that representations of goal and subgoal locations are elicited during path planning in lateral prefrontal cortex [3] and that this area guides multistep planning at the level of action goals and not motor actions [4]. A human study revealed the importance of striatum, medial temporal lobe and frontal cortex for navigation planning [5]. In rodents, model-based computations have been linked to a neuronal circuit involving the hippocampus and the ventral striatum [6]. Still, we have incomplete knowledge on (if and) how the brain implements model-based computations.

From a computational perspective, the benefits of goals and subgoals have been studied at three different timescales: learning, inference, and control. During *learning*, subgoals permit learning more efficiently by reducing the search space. This is well exemplified by *temporal abstraction* methods in hierarchical reinforcement learning (HRL) such as the Options framework [7]. Options can be conceptualized as sort of macro-actions whose termination conditions are subgoals. During *planning*, subgoals reduce the search space by permitting planning at a higher level of temporal abstraction (i.e., at the level of macro-actions), see also [8]. During *control*, subgoals permit maintaining the smallest possible information in working memory that is sufficient for task achievement [9] and supports efficient monitoring processes. Despite these progresses, we still lack an integrative theory of the computational role of subgoals in learning, inference, and control.

We argue that the main role of subgoals is reducing the complexity of learning, inference, and control tasks, by guiding the realization and selection of more compact control *programs*. A *program* can be defined as the sequence of actions necessary for the transition from an initial state s to a subgoal state sg. We assume that a program can be determined from a policy π if s and sg are known. Key to our formulation is the conversion of the length of a program (i.e., the number of actions necessary to reach sg from s) into a probability by following principles of descriptive complexity [10]. This approach formalizes the "Occam's razor" principle: a priori, among the strings that represent the procedures returning an output, "simpler" strings (i.e., strings with low descriptive complexity) are more probable. Our formalization uses information-theoretic measures based on Solomonoff's Algorithmic Probability and Kolmogorov Complexity [11, 12].

2 Methods and results

To test the hypothesis, we realized a Dynamic Bayesian Network (DBN) [13] that infers subgoals and policies by considering the descriptive complexity of the resulting programs. The inference uses the graphical model described in Fig. 1. In the model, the transition $P(\pi|SG, S)$ captures the concept of an Option but is expressed in a probabilistic way. Note that we focus on inference, not learning; for this reason the DBN structure and parameters are assumed to be known.

We cast planning and policies selection as *probabilistic inference* problems, see [14, 15, 16, 17, 18, 19, 20]. The inference follows the pseudocode of Algorithm 1. Intuitively, the goal of the inference is finding a policy running from the initial state s_t to a final goal state s_{goal}, which are assumed to be known. Although a policy can be found that covers the whole trajectory from s_t to s_{goal}, the resulting inference would be very costly and often infeasible for even moderately large state spaces. A useful solution in HRL is splitting the search into more manageable subgoals and corresponding Options. In our formulation, the choice of subgoals and policies is driven by considerations of minimum description length.

Algorithm 1 Pseudo-code of the inference procedure

$t = 0$
set S_0 to the starting state s_0
sample a subgoal state sg_0 from the prior probability distribution $p(SG_0)$
sample a policy π from the conditioned probability distribution $p(\Pi|sg_0, s_0)$
determine the action a_0 depending on π and s_0
set the termination condition state f_0 according to $p(F_0|sg_0, s_0)$
while $(F_t \neq 2)$ **do**
  $t = t + 1$
  determine the state s_t by means of $p(S_t|a_{(t-1)}, s_{(t-1)})$
  sample the state sg_t from the conditioned probability distribution $p(SG_t|f_{(t-1)}, sg_{(t-1)})$
  sample a policy π from the conditioned probability distribution $p(\Pi|sg_t, s_t)$
  set the action a_t depending on π and s_t
  evaluate the termination condition variable F_t according to $p(F_t|sg_t, s_t)$
end while

The inference uses the initial state s_t as a clamped (i.e., observed) state, and the goal state s_{goal} as a prior on the subgoal node SG (this value $P(s_{goal})$ is the only parameter of the model). Setting goals as priors distinguishes our approach from planning-as-inference methods and is similar to the *active inference* scheme of [21].

The inference also uses additional priors on SG that essentially indicate the more likely subgoals in the environment. In HRL the problem of finding useful subgoals for Options is widely debated; most studies have assessed that *bottlenecks* (e.g., a door in a house-like navigation domain) are often useful subgoals [22, 23]. To extract subgoals we used a method inspired by [9] that consides for each state "the amount of Shannon information that the agent needs to maintain about the current goal at a given state to select the appropriate action". We inflect this measure in a probabilistic way by computing the probability that a subgoal sg is the output of some program given each of state s and each of the policy π. Thus, the a priori probability of a generic state to be a subgoal depends on how many programs halt in that state and how long they are (see [11] for a similar method).

Because the number of policies to be searched in all except the most trivial environments is huge, we adopt a *sampling* method: importance sampling [11]. During the sampling, the probability of selecting a specific policy π depends on the length of the *program* that can be generated from π and that permits a transition from the currently examined state s and the currently examined subgoal sg. (Remember that a *program* is the sequence of actions necessary for a transition from s to sg; this length can be exploited to estimate the related a priori probability using the methods devised by Solomonoff and Kolmogorov.) This method returns pairs of subgoals and policies (or in other words, Options) that would (ideally) specify the shortest possible programs from the initial to the goal state. As we noted before, at least in principle this has benefits for both inference and control. During inference, this method permits searching through a smaller search space. During control, it permits achieving goals using the shortest path while at the same time having the smallest cognitive load (e.g., as shorter programs require fewer bits of information to be described, working memory load is alleviated).

The role of the node F is monitoring (sub)goal achievement and guiding the transitions between subgoals (see [20]). When the current state s is the same as the currently selected subgoal sg, a *rest policy* π_ε (i.e., a specific policy associating to every state a "rest" action ε) is selected. The node F thus determines the transition to a new subgoal (selected during inference and incorporated in the transition $SG_t \rightarrow SG_{t+1}$). When the final goal s_{goal} is reached, the transitions end.

We present simulated experiments in a "four-rooms" scenario (similar to [7]) aiming at comparing the performance of our method (which uses subgoals) with an equivalent one that does not use subgoals. In the comparison we consider the number of successfully reached goals, the optimality of behavior (expressed here as the length of the path to achieve a goal), the complexity of the inference (i.e., the number of inferential cycles necessary to infer a control policy) and the complexity of the control (i.e. the cumulative number of bits in working memory necessary to achieve the task, see [9]).

Fig. 1 shows the synthetic discrete world we used. It has 18 states $S = \{s_1 \ldots, s_{18}\}$ and is composed of four "rooms" with a single connection among them (S7 and S12). Even in this simple scenario, the number of potential policies is in around seven millions, making exact inference impracticable. We made 10 simulation runs per number of particles (50, 100 and 1000 particles) with two different modalities. In the first (*without-subgoals*) modality the probability of choosing a subgoal different from the goal state is zero ($P\left(SG \neq s_{goal}\right) = 0$). In the second (*with-subgoals*) modality a discrete probability distribution on the subgoals is used ($P\left(SG \neq s_{goal}\right) > 0$). In the experiments we assumed s_1 as starting state and s_{18} as goal (notice that the probability of choosing the goal state s_{18} is raised to make it the most probable state). The priors on subgoals (calculated using the aforementioned method inspired to [9]) are shown in gray scale in Fig. 1.

Fig. 2 shows the distribution of subgoals found by our inference procedure averaged on the different runs. Results show that in the tested environment, strategies including two subgoals before the actual goal (with a total of three subplans) are more successful than others. Successful examples include for example [S2, S3, S18] and [S16, S17, S18]. Tab. 2

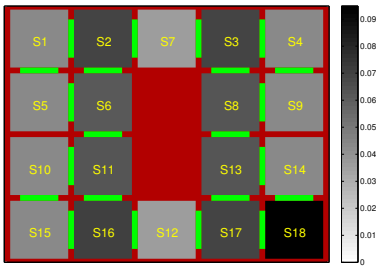


Figure 1: Environment representation with 18 states and subgoal priors depicted in gray scales (S18 is the goal state). Green and red bars represent doors and walls, respectively.

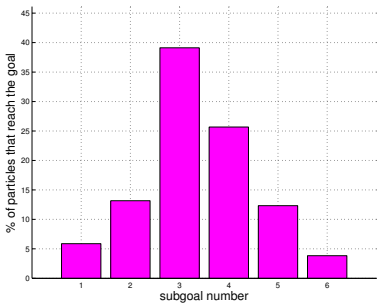


Figure 2: Mean subgoal distribution of the successful strategies. Note that the goal is included so a subgoal of 1 indicates that no additional subgoals were selected.

# of particles	% of success $P(SG \neq s_{goal}) = 0$	% of success $P(SG \neq s_{goal}) > 0$
50	29	45
100	33	47
1000	36	50

Table 2: Percentage of particles that correctly find a plan to the goal, for different number of particles (50, 100 and 1000).

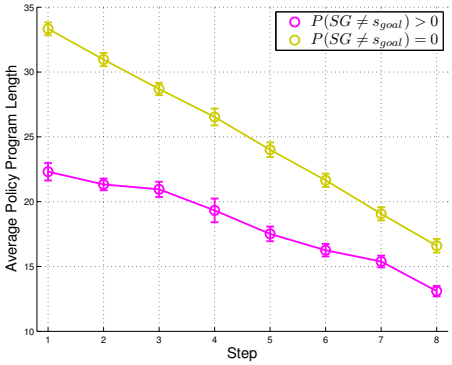


Figure 3: Average Program Length of policies per step for the two modalities of execution (standard deviation shown).

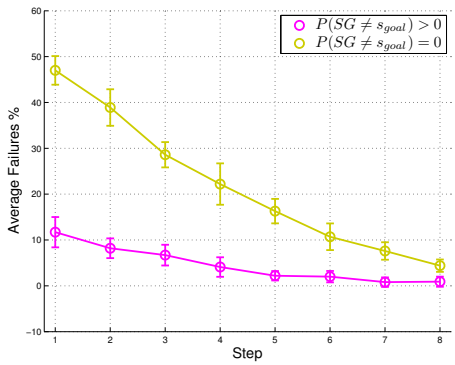


Figure 4: Average Failures Percentage of particles that do not represent a successful strategy (standard deviation shown).

shows the differences between the percentage of successful strategies carried out in the *without-subgoals* and *with-subgoals* modalities. The latter strategy achieves a better performance by using subgoals to split the search space.

Fig. 3 shows the average program length of the policies per step in the two modalities of execution (*with-subgoals* is pink, *without-subgoals* is yellow) for $N = 100$ particles and averaged on 10 runs (we obtained similar results with 50 and 1000 particles). This measure is related to the working memory necessary for inference and control, suggesting that the method using subgoals requires less memory resources. Fig. 4 shows the average percentage of particles that fail to find a suitable strategy (with $N = 100$). The results show that the inference method using subgoals is more efficacious, especially in the first steps. The percentage of failures is stable in all the steps, suggesting robustness of the method.

3 Conclusions

We proposed that goals and subgoals help lowering the description complexity of task-relevant information during learning, inference, and control. We presented preliminary evidence suggesting that model-based decision-making can use subgoals to lower the descriptive complexity of the planned policies and programs.

From the computational perspective, our proposal links to *planning-as-inference* [14, 15], which uses probabilistic inference to reach desired rewards [18] or goals [19, 20] and to *active inference* where goals are used as Bayesian *priors* in a variational probabilistic scheme that minimizes free energy [21]. Similarly, we use goal states as priors but we also consider subgoals and use descriptive complexity to evaluate candidate policies. Still, in large environments searching through all the possible policies is inefficient. This problem can be alleviated by assigning priors to policies (depending e.g., on past

searches or the average length of their associated programs) or "caching" them. This could permit at least in principle to form libraries of Options or skills that can be reused across families of problems, as in *transfer learning* [24]. A further implication of this method is that goals and subgoals guide information compression in the cortical hierarchies by biasing which control programs are stored. The idea that information compression is a key organizing principle brain hierarchies has received some attention in neuroscience [25] but its empirical validity remains to be tested.

Besides, our study can offer a normative perspective on planning and subgoal processing in living organisms. The monkey PFC encodes a sequence of activation of goals and subgoals during a delay period prior to action [4, 3] and monitors goals at feedback time [26]. The proposed model suggests a possible computational principle for the encoding and monitoring of subgoal sequences. Further evidence is necessary to assess the biological plausibility of the model.

References

[1] E. K. Miller and J. D. Cohen. An integrative theory of prefrontal cortex function. *Annu Rev Neurosci*, 24:167–202, 2001.

[2] Richard E Passingham and Steven P Wise. *The neurobiology of the prefrontal cortex: anatomy, evolution, and the origin of Insight*, volume 50. Oxford University Press, 2012.

[3] Naohiro Saito, Hajime Mushiake, Kazuhiro Sakamoto, Yasuto Itoyama, and Jun Tanji. Representation of immediate and final behavioral goals in the monkey prefrontal cortex during an instructed delay period. *Cereb Cortex*, 15(10):1535–1546, Oct 2005.

[4] Hajime Mushiake, Naohiro Saito, Kazuhiro Sakamoto, Yasuto Itoyama, and Jun Tanji. Activity in the lateral prefrontal cortex reflects multiple steps of future events in action plans. *Neuron*, 50(4):631–641, May 2006.

[5] Dylan Alexander Simon and Nathaniel D Daw. Neural correlates of forward planning in a spatial decision task in humans. *J Neurosci*, 31(14):5526–5539, Apr 2011.

[6] Matthijs A A. van der Meer and A.D. Redish. Expectancies in decision making, reinforcement learning, and ventral striatum. *Frontiers in Neuroscience*, 4:6, 2010.

[7] R.S. 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.

[8] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 220–229. Morgan Kaufmann Publishers Inc., 1998.

[9] Sander G van Dijk and Daniel Polani. Grounding subgoals in information transitions. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 105–111. IEEE, 2011.

[10] M. Li and P. Vitànyi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2nd edition, 1997.

[11] David J. C. Mackay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 1st edition, June 2002.

[12] Juergen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 10:10–5, 1997.

[13] Kevin P. Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.

[14] H. Attias. Planning by probabilistic inference. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[15] Matthew Botvinick and Marc Toussaint. Planning as inference. *Trends Cogn Sci*, 16(10):485–488, Oct 2012.

[16] Giovanni Pezzulo and Francesco Rigoli. The value of foresight: how prospection affects decision-making. *Front. Neurosci.*, 5(79), 2011.

[17] Giovanni Pezzulo, Francesco Rigoli, and Fabian Chersi. The mixed instrumental controller: using value of information to combine habitual choice and mental simulation. *Front Psychol*, 4:92, 2013.

[18] Alec Solway and Matthew M. Botvinick. Goal-directed decision making as probabilistic inference: A computational framework and potential neural correlates. *Psychol Rev*, 119(1):120–154, Jan 2012.

[19] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd international conference on Machine learning*, pages 945–952. ACM, 2006.

[20] Deepak Verma and Rajesh P. N. Rao. Planning and acting in uncertain environments using probabilistic inference. In *IROS*, pages 2382–2387. IEEE, 2006.

[21] Karl J Friston, Jean Daunizeau, and Stefan J Kiebel. Reinforcement learning or active inference? *PLoS One*, 4(7):e6421, 2009.

[22] M. Botvinick, Y. Niv, and A. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 119(3):262–280, 2009.

[23] Nir Lipovetzky and Hector Geffner. Width and serialization of classical planning problems. In *ECAI*, pages 540–545, 2012.

[24] G.D. Konidaris and A.G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

[25] S J. Kiebel, J. Daunizeau, and K J. Friston. A hierarchy of time-scales and the brain. *PLoS Comput Biol*, 4(11):e1000209, Nov 2008.

[26] Satoshi Tsujimoto, Aldo Genovesio, and Steven P. Wise. Frontal pole cortex: encoding ends at the end of the endbrain. *Trends Cogn Sci*, 15(4):169–176, Apr 2011.

Learning from demonstrations: Is it worth estimating a reward function?

Bilal Piot
MaLIS research group (SUPELEC)-UMI 2958 (GeorgiaTech-CNRS)
2 rue Edouard Belin, Metz, 57070 FRANCE
bilal.piot@supelec.fr

Matthieu Geist
MaLIS research group (SUPELEC)
2 rue Edouard Belin, Metz, 57070 FRANCE
matthieu.geist@supelec.fr

Olivier Pietquin
MaLIS research group (SUPELEC)-UMI 2958 (GeorgiaTech-CNRS)
2 rue Edouard Belin, Metz, 57070 FRANCE
olivier.pietquin@supelec.fr

Abstract

This paper provides a comparative study between Inverse Reinforcement Learning (IRL) and Apprenticeship Learning (AL) reduced to classification. IRL and AL are two frameworks for the imitation learning problem where an agent tries to learn from demonstrations of an expert. In AL, the agent tries to learn the expert policy whereas in IRL, the agent tries to learn a reward which can explain the behavior of the expert. Then, the optimal policy regarding this reward is used to imitate the expert. One can wonder if it is worth estimating such a reward, or if estimating a policy is sufficient. This quite natural question has not really been addressed in the literature so far. We provide partial answers, both from a theoretical and empirical points of view.

Keywords: Learning from demonstrations, Inverse Reinforcement Learning, Apprenticeship Learning.

Acknowledgements

The research leading to these results has received partial funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°270780.

1 Introduction

This paper provides a comparative study between two methods that attempt to solve the imitation learning problem where an apprentice tries to learn from demonstrations of an expert. These two methods are Apprenticeship Learning (AL) [1] and Inverse Reinforcement Learning (IRL) [7]. In AL, the agent tries to learn the expert policy or at least a policy which is as good as the expert policy (according to an unknown reward function). In IRL, the agent tries to learn a reward which can explain the behavior of the expert. Then, the optimal policy regarding this reward is used to imitate the expert. AL can be reduced to classification [5, 6, 9] where the agent tries to mimic the expert policy via a Supervised Learning (SL) algorithm. There exist also several AL algorithms inspired by IRL such as [1, 8] but they need to solve recursively MDPs which is a difficult problem (not considered in this paper) when the state space is large and the dynamics of the MDP is unknown.

First, we analyse the difference of value functions between the apprentice and the expert policies when a classifier is used as AL method (in the infinite horizon case). When compared to the sole (as far as we know) related result in IRL, quantifying the quality of an apprentice trained with the recently introduced Structured Classification based IRL (SCIRL) algorithm [4], this analysis tells us that estimating a reward only adds errors. Then, we perform an empirical study on the generic Garnet framework [2] to see if this first partial answer is confirmed. It turns out that it actually strongly depends on the (unknown) reward: roughly, the less informative the reward is, the more IRL provides gains compared to classification. Finally, we push this empirical study even further by perturbing the dynamics of the MDP, which goes beyond the studied theory. In this case, the advantage of IRL is even clearer.

2 Background and Notations

2.1 General notations and Markov Decision Process

Let $\mathcal{X} = (x_i)_{\{1\leq i\leq N_\mathcal{X}\}}$ be a finite set and $f \in \mathcal{R}^\mathcal{X}$, f is identified to a column vector and f^T is the transposition of f. The set of probability distributions over $\mathcal{X}$ is noted $\Delta_\mathcal{X}$. Let $\mathcal{Y}$ be a finite set, $\Delta_\mathcal{X}^\mathcal{Y}$ is the set of functions from $\mathcal{Y}$ to $\Delta_\mathcal{X}$. Let $\zeta \in \Delta_\mathcal{X}^\mathcal{Y}$ and $y \in \mathcal{Y}$, $\zeta(y) \in \Delta_\mathcal{X}$, which can be seen as the conditional distribution probability knowing y, is also noted $\zeta(.|y)$ and $\forall x \in \mathcal{X}, \zeta(x|y) = [\zeta(y)](x)$. Besides, let $A \subset \mathcal{X}$, then $\chi_A \in \mathbb{R}^\mathcal{X}$ is the indicator function on A. Moreover, let $\mu \in \Delta_\mathcal{X}$, $\mathbb{E}_\mu[f]$ is the expectation of f with respect to μ. Let $x \in \mathcal{X}$, $x \sim \mu$ means that x is sampled according to μ. Finally, we define $\|f\|_\infty = \max_{x\in\mathcal{X}} f(x)$.

A finite Markov Decision Process (MDP) is a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ where $\mathcal{S} = (s_i)_{\{1\leq i\leq N_\mathcal{S}\}}$ is the state space, $\mathcal{A} = (a_i)_{\{1\leq i\leq N_\mathcal{A}\}}$ is the action space, $\mathcal{P} \in \Delta_\mathcal{S}^{\mathcal{S}\times\mathcal{A}}$ is the Markovian dynamics of the MDP, $\mathcal{R} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is the reward function and γ is the discount factor. A stationary and Markovian policy $\pi \in \Delta_\mathcal{A}^\mathcal{S}$ represents the behavior of an agent acting in the MDP $\mathcal{M}$. The quality of π in the infinite horizon framework is quantified by the value function $v_\mathcal{R}^\pi \in \mathbb{R}^\mathcal{S}$ which maps to each state the expected and discounted cumulative reward for starting in this state and following the policy π afterwards: $v_\mathcal{R}^\pi(s) = \mathbb{E}[\sum_{t\geq 0} \gamma^t \mathcal{R}(s_t, a_t)|s_0 = s, \pi]$. A policy $\pi_\mathcal{R}^*$ is said optimal (according to $\mathcal{R}$) if its value function $v_\mathcal{R}^*$ satisfies $v_\mathcal{R}^* \geq v_\mathcal{R}^\pi$ for any policy π and componentwise. Let P_π be the stochastic matrix $P_\pi = (\sum_{a\in A} \pi(a|s)\mathcal{P}(s'|s,a))_{\{(s,s')\in\mathcal{S}^2\}}$, we write, when it exists, $\rho_\pi \in \mathbb{R}^\mathcal{S}$ the stationary distribution of the policy π (satisfying $\rho_\pi^T P_\pi = \rho_\pi^T$).

2.2 AL and IRL

In the AL framework, the apprentice, given some observations of the expert policy π_E, tries to learn a policy π_C which is as good as the expert policy according to the unknown reward $\mathcal{R}$. Thus, the apprentice tries to find π_C such that the quantity: $\mathbb{E}_\nu[v_\mathcal{R}^{\pi_E} - v_\mathcal{R}^{\pi_C}]$ is the lowest possible, where $\nu \in \Delta_S$. In general $\nu = \rho$ where ρ is the uniform distribution or $\nu = \rho_{\pi_E}$ (also noted ρ_E). In the IRL framework, the apprentice tries to learn a reward $\hat{\mathcal{R}}$ which could explain the expert behavior. More precisely, given some observations of the expert policy π_E, the apprentice learns a reward $\hat{\mathcal{R}}$ such that the quantities $\mathbb{E}_\nu[v_{\hat{\mathcal{R}}}^{\pi^*_{\hat{\mathcal{R}}}} - v_{\hat{\mathcal{R}}}^{\pi_E}]$ or $\mathbb{E}_\nu[v_\mathcal{R}^{\pi_E} - v_\mathcal{R}^{\pi^*_{\hat{\mathcal{R}}}}]$ are the lowest possible.

3 Theoretical study

3.1 AL reduced to classification for the infinite horizon case

A way to realize an AL method is by mimicry via classification. More precisely, we assume that some demonstrations examples $D_E = (s_i, a_i)_{\{1\leq i\leq N\}}$ where $a_i \sim \pi_E(.|s_i)$ are available. Without loss of generality, we assume that the states s_i are sampled according to some probability distribution $\nu \in \Delta_\mathcal{S}$. The data (s_i, a_i) are sampled according to the distribution μ_E such that: $\mu_E(s, a) = \nu(s)\pi_E(a|s)$. Then, a classifier is learnt based on these examples. This outputs a policy

$\pi_C \in \mathcal{A}^{\mathcal{S}}$, which associates to each state an action. The quality of the classifier is quantified by the classification error: $\epsilon_C = \mathbb{E}_{\mu_E}[\chi_{\{(s,a)\in\mathcal{S}\times\mathcal{A},\pi_C(s)\neq a\}}]$. The quality of the expert may be quantified with $v_{\mathcal{R}}^{\pi_E}$. Usually, it is assumed that the expert is optimal, but it is not necessary for the following analysis (the expert may be sub-optimal respectively to $\mathcal{R}$). The quality of the policy π_C can also be quantified by its value function $v_{\mathcal{R}}^{\pi_C}$. In the following, we bound $\mathbb{E}_\nu[v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}]$ which represents the difference between the quality of the expert and the classifier policy. If this quantity is negative, that is fine, because (in mean), π_C is better than π_E. So, only an upper bound is provided. This upper-bound shows the soundness of classification.

Let define the following concentration coefficient: $C_\nu = (1-\gamma)\sum_{t\geq 0}\gamma^t c_\nu(t)$ where $\forall t \in \mathbb{N}, c_\nu(t) = \max_{s\in\mathcal{S}} \frac{(\nu^T P_{\pi_E}^t)(s)}{\nu(s)}$. Notice that if $\nu = \rho_E$, which is a reasonable assumption, then $C_\nu = C_{\rho_E} = 1$.

Theorem 1 *Let π_C be the classifier policy (trained on the data set D_E to imitate the expert policy π_E). Let also ϵ_C be the classification error and C_ν the above defined concentration coefficient. Then $\forall \mathcal{R} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$: $\mathbf{E}_\nu[v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}] \leq \frac{2C_\nu\|\mathcal{R}\|_\infty}{(1-\gamma)^2}\epsilon_C$.*

The proof of Th. 1 is given on the supplementary material [1] and is based on the propagation of the classification error.

3.2 A bound on the finite-horizon case

In [9], the authors provide a bound for classification in the finite horizon case. They propose to learn, for each time $1 \leq t \leq H$ (H is the horizon), a classifier in order to build a non-stationary policy π_C imitating the expert. Doing so, they obtain the following theorem:

Theorem 2 (Syed 2010) *Let π_E be the expert non-stationary and Markovian expert policy and π_C the policy learnt by the H classifiers, then: $\mathbb{E}_\nu[v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}] \leq \min(2\sqrt{\epsilon_C}H^2, 4\epsilon_C H^3 + \delta_{\pi_E})\|\mathcal{R}\|_\infty$, where $\delta_{\pi_E} = \frac{\mathbb{E}_\nu[v_{\mathcal{R}}^* - v_{\mathcal{R}}^{\pi_E}]}{\|\mathcal{R}\|_\infty}$ represents the sub-optimality of the expert and ϵ_C the classification error.*

It is possible to compare these results with our bound, by informally noticing that the discount factor γ in the infinite horizon corresponds to an horizon of length $\frac{1}{1-\gamma}$: $\sum_{t\geq 0}\gamma^t = \frac{1}{1-\gamma}$. By replacing H by $\frac{1}{1-\gamma}$ in the the precedent bound, we obtain: $\mathbb{E}_\nu[v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}] \leq \min(\frac{2\sqrt{\epsilon_C}}{(1-\gamma)^2}, \frac{4\epsilon_C}{(1-\gamma)^3} + \delta_{\pi_E})\|\mathcal{R}\|_\infty$. So, if we informally identify the classification errors and the horizon H to $\frac{1}{1-\gamma}$, our bound is slightly better either by $\sqrt{\epsilon_C}$ or by $\frac{2}{1-\gamma}$. Moreover, as our bound is specific to the infinite horizon, it is more adapted to AL and IRL algorithms as most of them consider the infinite horizon case.

3.3 SCIRL and its performance bound

SCIRL uses the estimation of the expert feature expectation [4] as the basis function of a linearly parameterized score-based classifier. The classification error is noted ϵ_C and the reward outputted by the SCIRL algorithm is $\mathcal{R}_C$. Then, the performance bound for this algorithm is: $0 \leq \mathbb{E}_{\rho_E}[v_{\mathcal{R}_C}^* - v_{\mathcal{R}_C}^{\pi_E}] \leq \frac{C_f}{(1-\gamma)}\left(\frac{2\|\mathcal{R}_C\|_\infty\epsilon_C}{1-\gamma} + \bar{\epsilon}_Q\right)$, with $C_f = (1-\gamma)\sum_{t\geq 0}\gamma^t c_f(t)$ where $\forall t \in \mathbb{N}, c_f(t) = \max_{s\in\mathcal{S}} \frac{(\rho_E^T P_{\pi_{\mathcal{R}_C}^*}^t)(s)}{\rho_E(s)}$. The term $\bar{\epsilon}_Q$ is a measure of the error estimation of the feature expectation. This bound is specific to the reward $\mathcal{R}_C$ and the constant C_f is not equal to 1 when $\nu = \rho_E$, which makes this bound possibly quite worst than the classification bound, even when the expert feature expectation is perfectly estimated ($\bar{\epsilon}_Q = 0$) which is a strong assumption. This seems to indicate that this IRL algorithm is less interesting than a classification algorithm in theory. However, we will see, in Sec. 4, that for specific unknown rewards SCIRL can have much better performance than classification.

4 Empirical study

Here, experiments are conducted and show the interest of estimating a reward thanks to a general framework called the Garnet framework [2]. We choose it because the Garnets are finite MDPs with a tabular representation which allow comparing fairly the different approaches without the problem of bias induced by the choice of representation. The comparison is done between a classification algorithm and two recently published IRL algorithms: SCIRL and Relative Entropy IRL (RE) [3]. SCIRL and RE were chosen as benchmarks because they do not need to resolve iteratively MDPs which reduces the impact of Approximate Dynamic Programming (ADP). As they output a reward, the policy iteration algorithm is used to obtain a policy. These experiments show that the shape of the unknown reward, used to compute the expert policy, is crucial. Besides, we push this empirical study even further by perturbing the dynamics of the MDP and by showing that IRL provides a level of stability that no AL algorithms can reach for non-informative rewards.

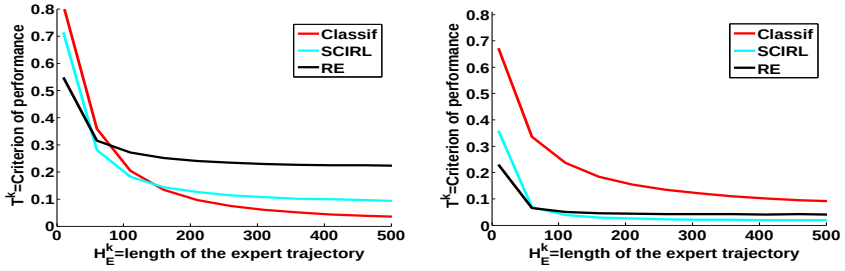
[1] `http://www.metz.supelec.fr/metz/personnel/piot_bil/RLDM.pdf`

4.1 Classification and IRL algorithms

The classification algorithm is an algorithm using a structured large-margin approach [6]. The second algorithm is SCIRL which is instantiated as described in [4]. The last algorithm is Relative Entropy (RE). It consists in minimizing the relative entropy between the empirical distribution of the state-action trajectories under a random policy and the distribution of the trajectories under a policy that matches the expert feature expectation [3].

4.2 The Garnet framework

The Garnet problems are a class of randomly constructed finite MDPs representative of the kind of finite MDPs that might be encountered in practice [2]. A stationary Garnet problem is characterized by 3 parameters and written as $Garnet(N_S, N_A, N_B)$. The parameters N_S and N_A are the number of states and actions respectively, and N_B is a branching factor specifying the number of next states for each state-action pair. The next states are chosen at random from the state set without replacement. The probability of going to each next state is generated by partitioning the unit interval at $N_B - 1$ cut points selected randomly. The reward $\mathcal{R}$ will be chosen depending on the experiments. Here, we compare the performances of the classification and the IRL algorithms. To obtain a generic result, we run the same experiment on 100 Garnets chosen randomly. In the first experiment, we choose a normally distributed reward, for each state-action couple, $R(s,a)$ is sampled according to a normal distribution of mean 0 and variance 1. This reward is very informative. For each Garnet and each length of trajectory $(H^k)_{\{1\leq k\leq 10\}} = (50, 100, ..., 500)$, we generate 100 expert trajectories. For each expert trajectory of length H^k, we run the three algorithms. The criterion of performance is $T^k = \frac{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi_E^p} - v_{\mathcal{R}}^{\pi_C^k}]}{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi_E^p}]}$, where π_E^p is the expert policy, π_C^k is the policy induced by the algorithm and ρ is the uniform distribution over the state space S. For each algorithm, we plot $(H_E^k, T^k)_{\{1\leq k\leq 10\}}$ where the performance has been averaged over expert trajectories of length H^k and all Garnets. Results are reported on Fig. 1(a). The second experiment is similar to the first one except



(a) Informative reward (b) Non-informative reward

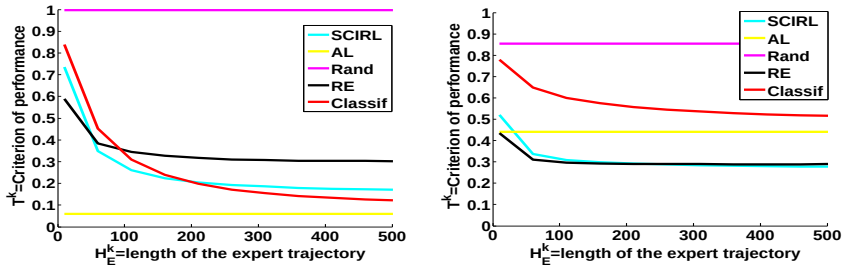
Figure 1: Garnets experiment

that the reward is non-informative. Indeed, we choose a sparse and state only dependent reward. Results are reported on Fig. 1(b). In Fig. 1(a), the classification algorithm has a better performance over the IRL algorithms. Indeed, as the reward is very informative, the choice of the action does not depend too much on the future states and the impact of the optimization horizon is strongly reduced. In Fig. 1(b), we see that the IRL algorithms work better than previously and the classification algorithm has its performance deteriorated. As the unknown reward is non-informative, the optimization horizon may be important and the classification is not able to take this temporal structure into account.

4.3 Dynamics perturbations

Here, we want to show that it can be interesting to retrieve the reward to be stable to dynamics perturbations. As the reward is seen as the most succinct hypothesis explaining the expert policy, we can expect that an optimal policy according to a reward outputted by IRL algorithms will be near-optimal even if there is a dynamics perturbation. The dynamics perturbations considered are the ones which keep identical the structure of the MDP. The structure of the MDP is, for a given state-action couple (s,a), the different states that could be reached by choosing the action a in state s, that is $\text{Supp}(P_{s,a})$ where $\text{Supp}(\nu)$ is the support of the distribution ν. The structure of the MDP is the set $(\text{Supp}(P_{s,a}))_{\{(s,a)\in\mathcal{S}\times\mathcal{A}\}}$ and a dynamic perturbation is the choice of a dynamics $\tilde{\mathcal{P}}$ different from $\mathcal{P}$ with the same structure. To obtain a generic result, we generate 100 Garnets as we have done previously. In the first experiment, we use the informative reward. For each Garnet, we compute the expert policy via the policy iteration algorithm. For each Garnet and each length of trajectory $(H^k)_{\{1\leq k\leq 10\}} = (50, 100, ..., 500)$, we generate 100 expert trajectories of length H^k. For each expert trajectory, we run the three algorithms. Then, for each Garnet, we generate 100 perturbed dynamics and we calculate, for each

trajectory of length H^k, the criterion of performance $T^k = \frac{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi^*} - v_{\mathcal{R}}^{\pi_C^k}]}{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi_E^p}]}$, where π^* is the optimal policy after perturbation, π_C^k is the policy induced by the algorithm and ρ is the uniform distribution over the state space S. For each algorithm, we plot $(H_E^k, T^k)_{\{1 \leq k \leq 10\}}$ where the performance has been averaged over all expert trajectories of length H^k, perturbed dynamics and Garnets. Results are reported on Fig. 2(a). The second experiment is similar to the first one except that



(a) Informative reward (b) Non-informative reward

Figure 2: Garnets experiment with dynamics perturbations

the reward is non-informative. Results are reported on Fig. 2(b). In Fig. 2, the curve named AL corresponds to the performance of the expert policy and the curve named Rand corresponds to a random policy. The curve named AL corresponds to the best achievable result of an AL algorithm as it is the performance of the expert policy in the original dynamics. In Fig. 2(a), the reward is informative, so a dynamic perturbation may not deteriorate too much the expert policy. Indeed, the impact of the optimization horizon must be small and the perturbation of dynamics will not change too much the optimal policy. We can observe this on Fig. 2(a), where the yellow curve noted AL is not so far away from 0. With this shape of reward, it is better to use a classification algorithm to have this stability property. However, in Fig. 2(b), as the reward is sparse, a dynamic perturbation leads to a deterioration of the performance of the expert policy. We see that IRL curves are under the yellow curve, which means that no AL algorithms will be able to reach that level of stability. Thus, estimating a reward function in that case is useful because it allows a stability that no AL algorithms is able to provide.

5 Conclusion and Perspectives

In this paper, we give some theoretical and empirical insights into the following question: is it worth estimating a reward function? In theory, we showed that there are no specific reasons to use IRL. However, the experiments conducted in this paper on a generic task show that for specific shapes of the unknown reward, IRL algorithms have better performances than classification. We think that the less informative the reward is, the bigger the impact of the optimization horizon is. This is a disadvantage for the classification method which does not take into account this horizon. Besides, when the dynamics is perturbed, the advantage is even clearer for IRL algorithms. However, there is no theoretical proof explaining why IRL work better with specific forms of rewards. This can be an interesting perspective to give more soundness to our experiments.

References

[1] P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. of ICML*, 2004.

[2] TW. Archibald, KIM. McKinnon, and LC. Thomas. On the generation of markov decision processes. *Journal of the Operational Research Society*, 1995.

[3] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *Proc. of AISTATS*, 2011.

[4] E. Klein, M. Geist, B. Piot, and O. Pietquin. Inverse reinforcement learning through structured classification. In *Proc. of NIPS*, 2012.

[5] D.A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document, 1989.

[6] N. Ratliff, J.A. Bagnell, and S.S Srinivasa. Imitation learning for locomotion and manipulation. In *Proc. of IEEE-RAS International Conference on Humanoid Robots*, 2007.

[7] S. Russell. Learning agents for uncertain environments. In *Proc. of COLT*, 1998.

[8] U. Syed and R.E. Schapire. A game-theoretic approach to apprenticeship learning. In *Proc. of NIPS*, 2008.

[9] U. Syed and R.E. Schapire. A reduction from apprenticeship learning to classification. In *Proc. of NIPS*, 2010.

Introspective Classification for Mission-Critical Decision Making

Rohan Paul, Hugo Grimmett, Rudolf Triebel* and Ingmar Posner
Mobile Robotics Group
University of Oxford, UK
{rohanp,hugo,hip}@robots.ox.ac.uk and rudolph.triebel@in.tum.de

Abstract

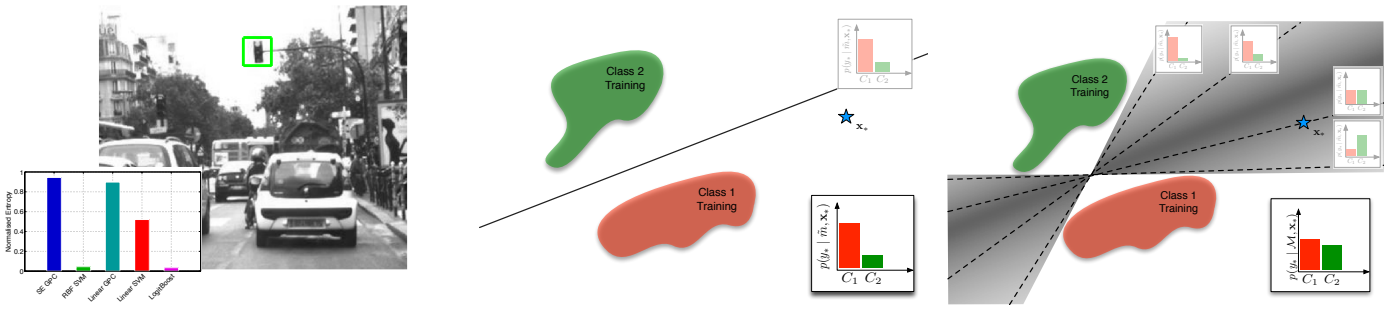
Classification *precision* and *recall* have been widely adopted by roboticists as canonical metrics to quantify the performance of learning algorithms. However, this paper advocates that for application domains which routinely require mission-critical decision making, such as robotics, good performance according to these standard metrics is desirable but *insufficient* to appropriately characterise system performance. We introduce and motivate the importance of a classifier's *introspective* capacity: the ability to mitigate potentially overconfident classifications by an appropriate assessment of how qualified the system is to make a judgement on the current test datum. We provide an intuition as to how this introspective capacity can be achieved and systematically investigate it in a selection of classification frameworks commonly used in robotics: support vector machines, LogitBoost classifiers and Gaussian Process classifiers (GPCs). Our experiments demonstrate that a framework such as a GPC exhibits a superior introspective capacity while maintaining commensurate classification performance to more popular, alternative approaches. We explore the benefits of an introspective classifier in the context of common robotics tasks such as classification, detection, and active learning for semantic mapping.

Keywords: life-long learning, classification uncertainty, semantic mapping

Acknowledgements

This work is funded under the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement Number 269916 and by the UK EPSRC Grant Number EP/J012017/1. We thank Prof. Paul Newman for his support.

*Dr. Rudolf Triebel is now at the Computer Vision Group, Technical University of Munich, Germany



(a) (b) (c)

Figure 1: (a) Classification uncertainty measured using normalised entropy for traffic light detectors based on five different classification frameworks applied to the window shown in green. All classifiers incorrectly label this window as *background*. However, the GPC variants do so with a significant amount of uncertainty while the others are inappropriately overconfident. Mission-critical decisions based on overconfident output will lead to catastrophic failure while an appropriately high amount of uncertainty when committing a mistake allows for remedial action to be taken. (b-c) An illustration of the two types of classification frameworks considered: (b) during training a *single* model is selected to classify an unknown datum $\mathbf{x}_*$; (c) training leads to a distribution over models which is considered entirely to arrive at the final prediction. This illustration is for the family of linear models, each annotated with its individual prediction and the overall predictive distribution (bottom right). The darker regions indicate higher probability weights associated with individual models. The overall predictive distribution in (b) stems from the single model used and is, in this case, inappropriately confident. In (c), the overall predictive distribution is moderated by computing the expectation over all models, resulting in a more appropriate uncertainty estimate — this is the introspective quality we seek.

1 Introduction

Classification based on a wide variety of sensor modalities has become an integral part of mobile robotics applications (see, for example, [1, 2, 3]). Often this is done with an implicit understanding that the application is agnostic to the classification method used. After all, for a number of classification frameworks the resulting *precision* and *recall*, quantities commonly used to characterise performance, are often commensurate across a wide variety of applications. We advocate that high precision and recall are desirable but do not suffice to fully characterise classification performance in robotics, where decisions taken are often mission or even safety critical. Crucially, this requires the classifier output to reflect an amount of uncertainty appropriate to a given situation. Even when hard class assignments are avoided by optimising an expected cost or reward, as is often the case in decision making, a *realistic* estimate of uncertainty when modelling the state of the world is pivotal; an autonomous car that misses a single traffic light with high confidence can suffer disastrous consequences (Fig. 1a).

We argue, therefore, that a classifier which is uncertain when it makes mistakes but certain when classification is correct, is more desirable than a classifier which makes correct and incorrect decisions with similarly high confidence. We investigate this *introspective* capacity in a number of classification frameworks commonly used in robotics: support vector machines (SVMs), LogitBoosting and Gaussian Process classifiers (GPCs). While we investigate specifically class-action, detection, and active learning scenarios, our treatment and findings apply to any aspect of robotics (and beyond) where action is required based on inference driven by raw sensor data. In this work, we present an overview of our two recent papers [4, 5], discussing key theoretical ideas and experimental results.

2 Introspection and Uncertainty

The introspective capacity of a classifier characterises its ability to *realistically* estimate the uncertainty in its predictions. The concept is motivated by the desire to take appropriate action when a classifier indicates high uncertainty. Our approach to introspection is grounded in the fact that the often cited assumption of independent and identically distributed (*i.i.d.*) training and test data is routinely violated in robotics: in the limit of continuous operation in the real world, one-shot classifier training is unlikely to be performed on a complete (or even fully representative) set of data.

Formally, let a classifier map an input $\mathbf{x} \in \mathbb{R}^d$ to one of a set of classes $C = \{C_1, \ldots, C_{|C|}\}$ via an associated label $y \in C$. Prior to training, domain specific knowledge is often used to constrain the family of classification models employed (e.g., in the form of a kernel, a covariance function or a type of base classifier). Classifier training then involves learning a set of (hyper-) parameters given a training dataset $\{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{X}|}\}$ denotes the set of feature vectors and $\mathcal{Y}$ denotes the corresponding class labels. The training data implicitly give rise to a distribution over the set of all possible models within the chosen family, $\mathcal{M}$, such that

$$\{\mathcal{X}, \mathcal{Y}\} \rightarrow p(m \mid \mathcal{X}, \mathcal{Y}), \quad m \in \mathcal{M}. \tag{1}$$

With a slight abuse of notation, m here denotes any member of the family of possible models, $\mathcal{M}$. In reality it is a function of the datum evaluated. In the following we make this relationship explicit by conditioning on both a model (or family of models) as well as on a test datum $\mathbf{x}_*$. Typically, training leads to the selection of a *single* model, $\tilde{m}$ from $\mathcal{M}$ such that a prediction y_* for a new, unseen feature vector $\mathbf{x}_*$ is obtained by approximating

$$p(y_* \mid \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_* \mid \tilde{m}, \mathbf{x}_*), \quad \tilde{m} \in \mathcal{M}. \tag{2}$$

This is illustrated in Fig. 1b. Common examples of this type of classification framework include SVMs and Boosting classifiers, where an optimisation is performed to select the best model given the training data. The *i.i.d.* assumption here is inherent since it is assumed that $\tilde{m}$ remains the best model for all predictions of unseen data. Breaking this assumption therefore often renders the chosen model suboptimal. An alternative to the single model approach are classification frameworks which take into account the *entire set* of possible models in the specified family, such that

$$p(y_* \mid \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_* \mid \mathcal{M}, \mathbf{x}_*). \tag{3}$$

This case is illustrated in Fig. 1c. Here the shading indicates the distribution $p(m \mid \mathcal{X}, \mathcal{Y})$ with darker shades indicating increased probability. To aid intuition, predictions of four randomly selected members of $\mathcal{M}$ are also illustrated. Final predictions are made by taking into account opinions from all members of $\mathcal{M}$, often via the computation of an expectation such as for a GPC (or its sparse variants) [6, 7]. Crucially, when considering an expectation over all of $\mathcal{M}$, the increased variance in feasible (and therefore likely) models at a distance from the training data leads to a moderation of the class predictions. This is the introspective quality we seek. In order to characterise the introspective capacity of a classification framework a transferable measure of the inherent uncertainty in the classification output is required. Here, we use an information-theoretic quantity known as normalised entropy, H_N, defined as

$$H_N \quad = \quad -\sum_{C_i \in C} p(y = C_i \mid \mathbf{x}) \log_{|C|} \left[p(y = C_i \mid \mathbf{x}) \right]. \tag{4}$$

This is equivalent to the Shannon entropy measure normalised by its maximum, which is the entropy of the $|C|$-dimensional uniform distribution, $\log(|C|)$. The result is an empirical measure ranging between 0 and 1 where a *higher* value indicates *greater* uncertainty in the classifier's belief.

3 Experimental Results

Our experiments investigate the introspective capacity of common classification frameworks (SVMs [8], LogitBoosting [9] and GPCs [6, 13] with linear and squared exponential (SE) kernel types where appropriate) in an autonomous driving and mapping setting. We focus on the *classification* of road signs and the *detection* of traffic lights. The classification scenario addresses the case where a decision is made between two, well-defined classes (e.g. two types of traffic signs) and investigates classifier performance as a third, previously unseen class is presented. The detection case involves separating a single class from a broad (in terms of intra-class variation) *background* class. Here, it is inherently assumed that the data representing the background class are sufficiently representative to capture any non-class object likely to be encountered, an assumption that may not hold true in practice, leading to mis-classifications. Further, we investigate the performance of a GP-based sparse introspective classifier in an active learning setting. We evaluate the performance in terms of the learning rate, data selection strategy and classification performance applied to the traffic light detection task and contrast against the commonly used SVM classifier (calibrated to provide probabilistic output).

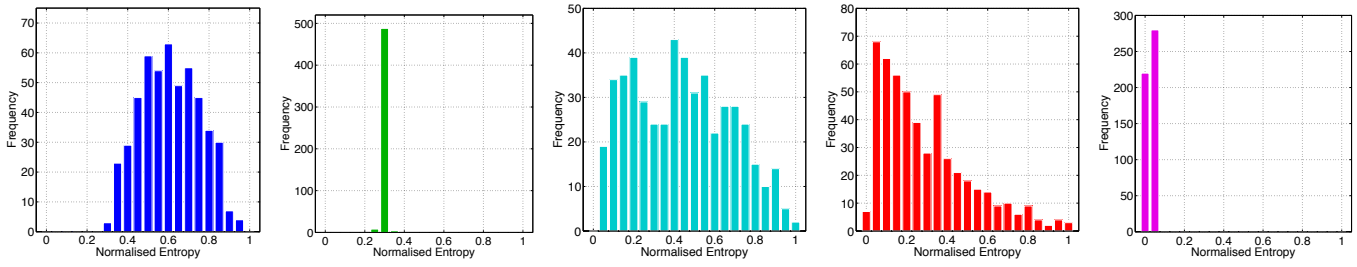
For the experiments we leverage two publicly available data sets: (i) German Traffic Sign Recognition Benchmark (GTSRB) dataset [10], which comprises over $50,000$ loosely-cropped images of 42 classes of road signs, with associated bounding boxes and class labels and (ii) Traffic Lights Recognition (TLR) data set [11], which comprises 11,179 colour images taken at 25 Hz from a car driven through central Paris at speeds under 31 mph with ground-truth labels for associated traffic light positions. We compute a template-based feature set inspired by Torralba *et al.* [12] in which a dictionary of partial templates is constructed, against which test instances are matched. For any given test instance, the normalised cross-correlation is computed for each dictionary feature.

3.1 Introspection in Classification and Detection

This section investigates classification output when a *third*, previously *unseen* class is presented to the classifier. We used the GTSRB road signs data set and arbitrarily selected two classes for training: *stop* and *lorries prohibited*. The classifiers were trained separating these two classes using a balanced training set of 400 data (200 per class) and applying a canonical training procedure for each classifier type, including five-fold cross-validation where appropriate. Classifier performance was evaluated using standard metrics on a hold-out set of another 400 class instances (200 of each class) of the same two classes. The f_1-performance was obtained as: SE GPC (0.995), SESE SVM (0.997), Linear GPC (0.995), Linear SVM (0.995) and LogitBoost (0.982). Classification performance is commensurate across all classifiers.

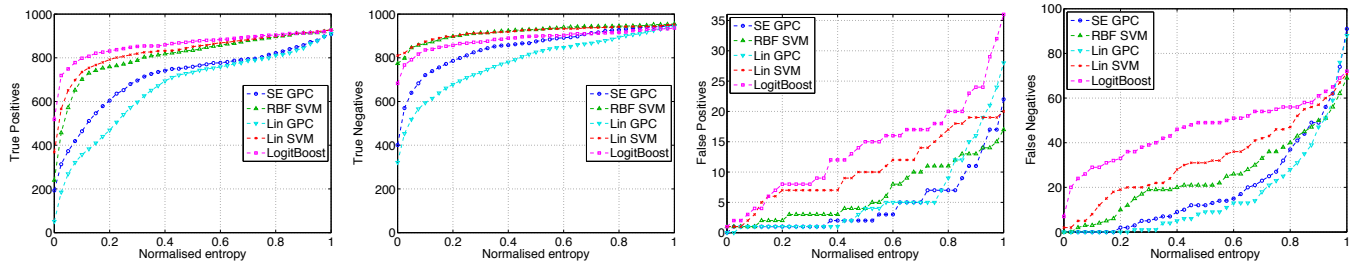
The classifiers are next retrained using the full 800 training data (400 per class) and the same canonical training procedures. They are then applied to 500 instances of the previously unseen class *roadworks ahead*. The resulting normalised entropy histograms are shown in Fig. 2. The mean normalised entropies for the GPC-based classifiers are significantly higher than those of the other classification frameworks, indicating that the the GPC-based classifiers exhibit greater uncertainty in their judgement. Conversely, the SE SVM and the LogitBoost classifier are extremely confident in their classifications with a very narrow distribution around a relatively low value of normalised entropy. This was an effect consistently observed throughout our experiments (see [4] for more detail), which we attribute to the relatively gradual decay of the estimated class posterior probabilities through feature space often encountered far away from the decision boundary. Features from an unseen class which are located in feature space at a distance from the decision boundary therefore only span a very narrow range of estimated class posterior probabilities.

Next, we investigate the same classification frameworks as before on the task of traffic light detection using the TLR data set. We split the dataset into two parts (at frame $7,200$ of $11,178$), with an approximately equal number of remaining labels in each part and



(a) SE GPC (b) SE SVM (c) Linear GPC (d) Linear SVM (e) LogitBoost

Figure 2: Normalised entropy histograms of the marginal probabilities for five classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on 500 instances of the unseen class *roadworks ahead*. Higher normalised entropy implies more uncertainty in classifier output. Note that the mean normalised entropy for the SE GPC is higher than that of the others.



(a) (b) (c) (d)

Figure 3: Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against normalised entropy. The classifiers have been trained on 500 traffic lights against 500 background patches, and tested on 1,000 instances of each. Note that lower normalised entropy implies more certainty in classification. A more introspective classifier is one that exhibits higher uncertainty (larger normalised entropy in its output) when processing difficult instances. Consequently, class decisions on output above a given normalised entropy threshold are deferred since the output is deemed ambiguous. This is desirable since a single bad decision can have disastrous consequences.

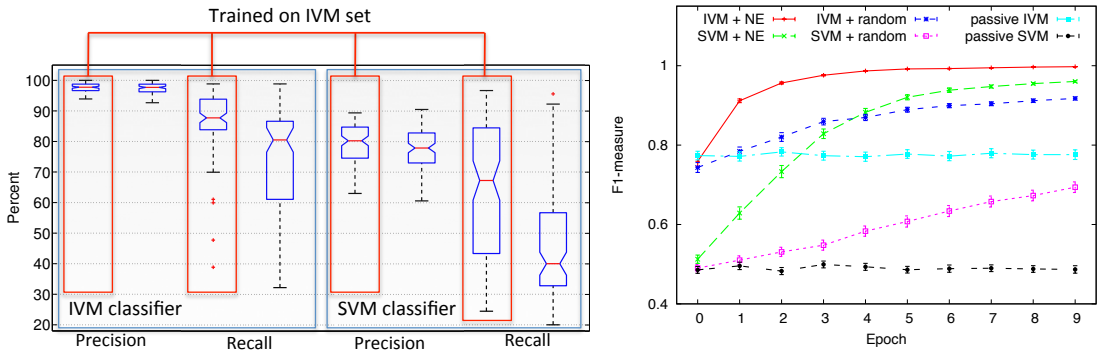
with no physical traffic lights in common. Positive data are extracted as labeled. Negative *background* data are extracted by sampling patches of random size and position from scenes in the dataset while ensuring that the patches do not overlap with positive instances. The data are then split into training $(1,000)$ and hold-out test set $(2,000)$. The f_1-performance was obtained as: SE GPC (0.941), SE SVM (0.956), Linear GPC (0.940), Linear SVM (0.953) and LogitBoost (0.945). As before, classification performance according to conventional metrics is commensurate across all frameworks.

Figure 3 illustrates how the lack of introspection can impact classification performance when accept/reject decisions are guided by classification confidence. Specifically, we show the *cumulative* effect of accepting classifications below a given uncertainty threshold. First we note that when classifications are accepted at any level of uncertainty (i.e. up to and including unity normalised entropy) all classification frameworks are commensurate in terms of true positives and true negatives (Fig. 3a-b). However, true positive and negative classifications occur generally at higher certainty (i.e. as normalised entropy tends to zero) for SVMs and LogitBoost classifiers than for the GPC variants. The latter are overall less certain about a significant number of correct classifications. Figure 3c-d, indicate that SVMs and LogitBoost classifiers are also significantly more confident when *misclassifying* data (an example of this is also shown in Fig. 1a). Significant numbers of mistakes are made at relatively low normalised entropy thresholds. The GPC variants, in contrast, accumulate comparable numbers of classification errors only at higher normalised entropy thresholds. The trade-off for this more realistic classification uncertainty assessment is a reduction in immediately correct classifications above the normalised entropy threshold. Note that this does not mean that these samples are misclassified. It only implies that some other remedial action might be taken — for example obtaining label confirmation from a human operator.

3.2 Active learning and the Benefit of Introspection

When confronted with uncertainty in classification output, a common remedial action consists of seeking human assistance to disambiguate the true class label. The new class information for uncertain data can be used to re-train or evolve the classifier leading to an active learning approach. Next, we apply a GP-based introspective classifier to active learning in the context of traffic light detection from a mobile platform. For efficiency and scalability to large data sets, we use a sparse GPC-based model, the Informative Vector Machine (IVM) [13]. The IVM uses information theoretic sparsification yielding scalability to large data sets while retaining its introspective nature via a marginalization over models induced by its non-parametric formulation.

We evaluate and compare the classification accuracy, learning rate and the selection of uncertain points for disambiguation for the IVM and the more commonly used SVM classifiers. For details please see [5]. The TLR data set is split into 10 epochs, each consisting of a training phase, a classification phase, and a feedback phase. During each classification phase, the classifiers are then



(a) (b)

Figure 4: The benefit of introspection. (a) Data selected by an introspective classifier lead to an improved learning rate in terms of precision-recall for both IVM and SVM over that selected by a non-introspective classifier. (b) Classification performance and (learning rates) for both IVM and SVM variants as indicated by the f_1-measure after each epoch (avg. over 100 experimental runs). The IVM using a normalised entropy-based data selection strategy (IVM+NE) consistently outperforms all other active learning variants in terms of overall performance and learning rate.

tested on a batch of 1,000 points randomly drawn from the test set. Then the 10 points with the highest normalised entropy (providing they are over a threshold empirically set to be 0.97) are greedily added to the training set, ready for retraining at the start of the next epoch. Note that each classifier (IVM and SVM) makes its own choices regarding which points to add for the next epoch.

We evaluate whether the use of an introspective classifier leads to more informative questions being asked when selecting data for human labelling and inclusion into the training set. Both an IVM and an SVM are initially trained on the same data. Then, 1,000 new data are shown to both classifiers for testing. Each chooses 10 data to add to the training set for the next round, resulting in *two* new and different training sets: the 'IVM set' and the 'SVM set'. A new IVM and SVM are now trained on *each* of the two new sets and evaluated on a further 1,000 held out data points. This process thus gives rise to four classifiers: two IVMs trained on data selected by an IVM and a SVM respectively, and two equivalent SVMs. We compute precision and recall for all four classifiers. The results after 100 repetitions of this experiment are shown in Fig. 4a. As expected, both the IVM and the SVM perform better when trained on the dataset chosen, introspectively, by the initial IVM, suggesting that the questions asked by the IVM tend to be significantly more informative. The overall effect of introspection in an active learning setting is therefore an increased learning rate, as shown in Fig. 4b, where the IVM learner based on a normalised entropy selection policy outperforms the equivalent SVM based method both in absolute terms per epoch as well as in terms of relative increase (information gained) per epoch.

References

[1] B. Douillard, D. Fox, and F. Ramos, "Laser and vision based outdoor object mapping," in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[2] O. Ḿartínez-Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard, "Supervised semantic labeling of places using information extracted from sensor data," *Robot. Auton. Syst.*, vol. 55, no. 5, pp. 391–402, 2007.

[3] I. Posner, M. Cummins, and P. Newman, "A generative framework for fast urban labeling using spatial and temporal context," *Autonomous Robots*, 2009.

[4] H. Grimmett, R. Paul, R. Triebel, and I. Posner, "Knowing When We Don't Know: Introspective Classification for Mission-Critical Decision Making," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[5] R. Triebel, H. Grimmett, R. Paul, and I. Posner, "Introspective Active Learning for Scalable Semantic Mapping," in *Proceedings of Robotics: Science and Systems (RSS) Workshop on Active Learning in Robotics: Exploration, Curiosity and Interaction*, 2013.

[6] C. Williams and C. Rasmussen, "Gaussian processes for machine learning," 2006.

[7] A. Naish-Guzman and S. Holden, "The Generalized FITC Approximation," in *Advances in Neural Information Processing Systems*, 2007, pp. 1057–1064.

[8] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1.

[9] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[10] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012.

[11] R. C. of Mines ParisTech, "Traffic lights recognition (TLR) data set," http://www.lara.prd.fr/benchmarks/trafficlightsrecognition.

[12] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing Visual Features for Multiclass and Multiview Object Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 854–869, 2007.

[13] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," *Advances in Neural Information Processing Systems*, vol. 15, pp. 609–616, 2002.

Metric Learning for Invariant Feature Generation in Reinforcement Learning

Evan Kriminger, Austin J. Brockmeier, Luis G. Sanchez Giraldo, and Jose C. Principe
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611
http://www.cnel.ufl.edu/

Abstract

The success of reinforcement learning in real-world problems depends on careful selection of features to represent the state. Proper feature selection results in the increased ability to approximate the value function and in quicker learning, as only relevant information is emphasized. It is desirable to generate such features automatically, as this would otherwise be a trial and error process requiring a human expert. We propose a method to automatically map states to feature vectors, which are not only sufficiently descriptive of the environment, but are invariant to information not relevant to the agent's goal. The mapping is based on the principle that if the Q-values for two states under the same action are similar, then the states themselves should be similar. This similarity is realized in a space defined by a metric computed using an information-theoretic approach to metric learning. Our method works in conjunction with a Q-learning algorithm of choice and is suitable for large and continuous state spaces. We test our algorithm on a non-sequential decision task in which the state is an image. While the problem is in fact linear, our method greatly outperforms linear Q-learning.

Keywords: Reinforcement learning, automatic feature generation, metric learning

1 Introduction

For many practical problems, the number of states in the environment is too large for a reinforcement learning agent to visit all of them. This is not only true in the case of a continuous state space, where there are a continuum of states, but even in discrete spaces where the number of states grows exponentially with the size of our state vector. This is the well-known "curse of dimensionality" [1]. In such cases, function approximation techniques are necessary to predict the value function for states that have not yet been visited. In this approach, a mapping from a feature vector representing the state to the value function is learned, as the agent interacts with the environment. The choice of the features is therefore a crucial factor in the success or failure of reinforcement learning. In RL problems each state is associated with raw measured data about the environment. This data might be the pixels of an image for instance. The feature vector is mapped from this raw data and should effectively represent the state in such a way that is invariant to distracting elements. For instance, when a baseball is approaching, the action of catching it is the same whether it occurs in a baseball stadium or in a backyard. Successful feature selection enables us to recognize the similarity between the two situations, despite the fact that the background in each case is completely different. While recognizing such invariants drastically reduces the size of the state space, the feature vector must be sufficiently descriptive such that the agent is able to learn complicated behaviors. In the first implementation of TD-gammon [2], the state of a game of backgammon was represented by a vector consisting of 198 elements. Even if the features intuitively capture the relevant information of the problem, it may not be the best way to represent the state to the function approximator that is learning the value function. Trial and error may be required to find a representation that leads to success.

Choosing the state representation is left to the 'designer', but in real-world problems this is unsatisfactory. Generating the feature mapping automatically, is a necessary step in extending RL to more complicated problems and to domains where experts cannot hand pick features. The literature on automatic feature generation is relatively sparse, owing partially to the difficulty of the problem. Many of the existing methods base the learning of features on the Bellman error of the value function estimates. These methods are known as Bellman Error Basis Function (BEBF) [3]. Other methods require estimates of information about the Markov Decision Process, such as the transition or adjacency matrix [4], which are difficult to acquire in many large problems. Additionally, most automatic feature generation methods are set in the domain of policy evaluation. Feature generation for control methods, such as Q-learning, is an even less studied problem. One such method [5] learns a metric for comparing states, such that states with similar transitions under the same action are close with respect to this metric. In this approach, two states x_i and x_j, at times i and j, respectively, are similar if their single step increments are similar, i.e. if $x_{i+1} - x_i$ and $x_{j+1} - x_j$ are similar. A potential problem with this method is that the differences between consecutive states may not reliably indicate similarity of states with respect to the goal of the agent. If distracting features are dominant, such as in visual systems, then the features of interest are not properly accounted for in the state transitions.

In this paper, we present an automatic feature generator for RL that is also based on metric learning. However, in our approach, the similarity between states is established with respect to the goal of our learning system, rather than temporal differences in the state vectors. We first present our method and the information-theoretic metric learning algorithm it utilizes. We then demonstrate the ability of our method to generate invariant features in a non-sequential decision problem, where the state is a black and white image.

2 Metric Learning for Reinforcement Learning

To create a feature representation that is invariant to information not relevant to the current decision problem, we use the following principle:

Two states are similar if the value of these states are similar under the same action.

Intuitively, the value function (specifically the Q-value) reflects the performance goals for the given problem independently of the state. If $Q(x, a) = Q(y, a)$ for states x and y and action a, then these states are similar, in the same way that a stadium and a backyard are similar when moving your arm to catch a baseball.

Given this principle, the features we choose to represent the state should exist in neighborhoods defined by their Q-values, not based on the Euclidean distances between the raw vectors that are associated with each state. The prospect of learning an approximation of Q-values supports the principle we are using. If we choose a representation where states with similar Q-values are clustered, the mapping from state/action pairs to Q-values can be smooth. In a representation where neighboring states may have wildly different Q-values, it may be difficult or even impossible to learn such a mapping. Imagine our baseball player is attempting first to catch a fly ball and then to field a ground ball. While the field looks identical in both cases, save for the small speck that is the ball, to hold his glove high in the air would be successful in the first case, and a failure in the second.

Consider a reinforcement learning agent that has interacted with the environment for N time steps to acquire tuples, $(x_i, a_i, Q(x_i, a_i))$, consisting of the state, action, and Q-value at each time i. We wish to group states based on the action

that was performed in that state, as these are the states for which we can compare the Q-values. Let $\mathcal{X}_a := \{x_i | a_i = a\}$, be the set of states for action a. Each pair of states in $\mathcal{X}_a$, for each a, provides a constraint on a metric m, such that if

$$|Q(x_i, a) - Q(x_j, a)| < |Q(x_i, a) - Q(x_k, a)| \quad \text{for } x_i, x_j, x_k \in \mathcal{X}_a,$$

then $m(x_i, x_j) < m(x_i, x_k)$. This is equivalent to considering a constraint on the mapping, ϕ, from states to feature vectors. In the latter case, $m(\phi(x_i), \phi(x_j)) < m(\phi(x_i), \phi(x_k))$, where m is simply the Euclidean distance. Learning the metric m or the mapping ϕ, that satisfies (or best satisfies) these constraints, provides the desired feature representation for our states.

Luckily, learning a metric is a well-investigated problem, as finding a 'good' similarity measure is common in pattern recognition and machine learning. The approach we consider is similar to metric-learning for classification [6, 7]. The concept of metric-learning is to parametrize a distance function such that states with similar Q-values are deemed close and states with very different Q-values are considered far apart. Most metric learning algorithms require hard information representing class membership. In our problem, the information in the form of the Q-values is soft, and we must find a metric on states which parallels the distances between Q-values. For this reason, we learn the metric via an information-theoretic optimization problem that optimizes the information between the state representation and the Q-values [8].

2.1 Distances and Similarity

The similarity between samples x and x' on the ith dimensions is $\kappa(x^{(i)}, x^{(i)\prime}) = \exp(-\theta d(x^{(i)}, x^{(i)\prime})^2)$, where $d(\cdot, \cdot)$ is the Euclidean distance and θ is a kernel size parameter. Changing the kernel size adjusts how close the samples must be in order to be considered similar.

In terms of a group of samples, the pairwise distance matrix for the ith dimension is denoted D_i where $(D_i)_{j,k} = d(x_j^{(i)}, x_k^{(i)})$ for $j, k \in \{1, \ldots, n\}$. Likewise, the corresponding kernel matrix with kernel size parameter θ is $K_i = \exp(-\theta D_i^2)$.

A similar quantity can be defined for the similarity between Q-values and the corresponding kernel-matrix is denoted L.

2.2 Entropy

Rényi's α-order entropy is an information measure for probability distributions. Recent work [8], has shown how a similar quantity can be defined in terms of the eigenvalues of a positive definite kernel matrix. Using the formulation of Rényi's entropy on the eigenvalues yields a matrix-based analog to entropy [8]:

$$S_\alpha(B) = \frac{1}{1-\alpha} \log\left[\operatorname{tr}(B^\alpha)\right]. \quad (1)$$
$$\text{where } \operatorname{tr}(B) = 1$$

Unlike standard approaches that require knowing the distributions of the data or estimating its density using methods such as Parzen windows, this approach directly estimates an entropy quantity without ever requiring an explicit density function. Another key benefit is optimization can be done in terms of the kernel and distance matrix using matrix calculus.

From this a measure of conditional entropy $S_\alpha(B|C) = S_\alpha(B, C) - S_\alpha(C)$ can be applied, and this form of conditional entropy was used in previous work for learning a Mahalanobis distance [8].

2.3 Tensor Product Kernel

The tensor product between kernel functions is a joint measure for multivariate data formed as the product of kernels for each dimension of the input.

Considering the Gaussian kernel, the tensor product kernel corresponds to using a non-negative combination of the different Euclidean metrics as the argument to the kernel function. (A series of Hadamard products is denoted $\prod_{i=1}^N A_i = A_1 \circ \cdots \circ A_N$, and entry-wise power as $A_i^{\circ r}$)

$$K_\theta = \prod_i K_i^{\circ \theta_i} = \prod_i \exp(-\theta_i D_i^2) = \exp(-\sum_i \theta_i D_i^2)$$

Adjusting the parameters of $\theta \geq 0$ changes the kernel size of each component kernel, and this is equivalent to scaling each dimension of the input so as to form a new metric $d_\theta^2(x, x') = \sum_i \theta_i d^2(x^{(i)}, x^{(i)\prime})$.

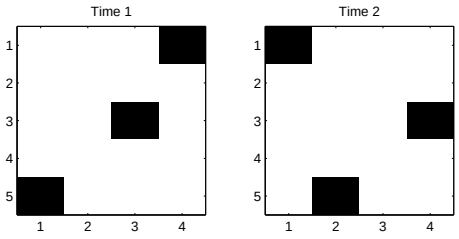


Figure 1: Example images for two consecutive time steps of the game.

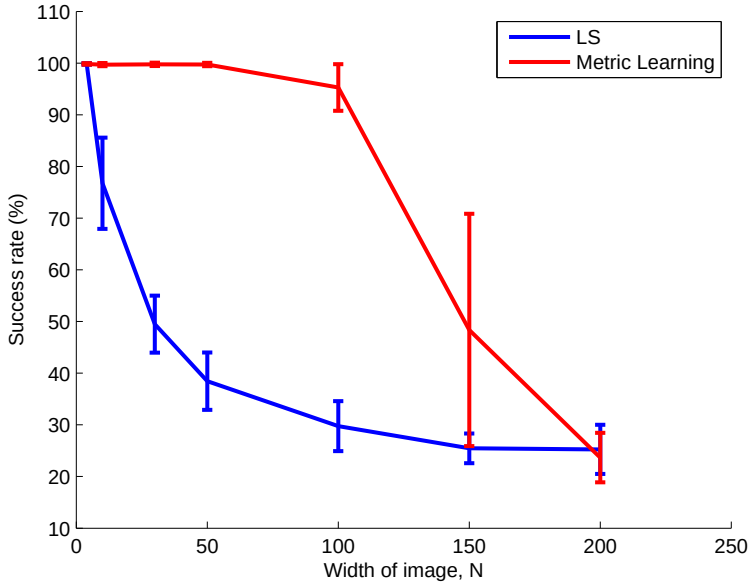


Figure 2: Success rate for least squares and metric learning methods. The error bars represent the interquartile range.

In order to learn this metric we consider the following information-theoretic optimization problem:

$$\begin{aligned} &\underset{\theta \geq 0}{\text{minimize}} \;\; S_\alpha(L, K_\theta) \\ &\text{subject to} \quad S_\alpha(K_\theta) = \eta \end{aligned} \tag{2}$$

This problem attempts to maximize the joint entropy of the state representation and the Q-values while constraining the marginal entropy. Allowing a large value of η allows the entropy of the data sample to increase, which is usually important for regularizing statistical estimates. However, if η is too high, a trivial solution is found where every sample is deemed different. A good choice of η is Rényi's entropy of the Q-values.

The relationship with the conditional entropy ($S_\alpha(L|K) = S_\alpha(L, K) - S_\alpha(K)$) can be seen by transforming the constraint into the Lagrangian formulation

$$\underset{\theta \geq 0}{\text{minimize}} \;\; S_\alpha(L, K_\theta) - \lambda \left(S_\alpha(K) - \eta \right) \tag{3}$$

with the Lagrange multiplier λ. However, the constraint on the non-negativity of the coefficients remains. As a simple approach, an unconstrained optimization is formed by re-parametrizing the function in terms of w where $\theta_i = 10^{w_i}$. By solving this optimization problem, a new metric can be learned as a linear combination of metrics.

3 Experiment

To test the ability of our generated features to represent the state, we consider a non-sequential decision task, in which the the state is a black and white image. By choosing a non-sequential task, the value of each action is known perfectly. This was done to isolate the problem of feature generation from value estimation. This problem consists of a image that is 5 pixels in height and N pixels wide. At most one pixel of each column is black, while the rest are white. At each time step, the black pixels shift one to the right, and with probability 0.5, a randomly selected row of the first column is set to black. When a pixel in the rightmost column is black, our RL agent must select the row in which it exists; thus there are 5 actions. If the agent selects the black pixel, a reward of +1 is received. If it misses the black pixel, a reward of -1 is received. If all pixels in the rightmost column are white, the reward is 0 for all actions. The reward in this case is the Q-value, as this is a non-sequential problem. See Figure 1 for an illustration of the game.

The state in this game is the image itself, which takes the form of a vector with $5 \cdot N$ binary elements, representing the values of each pixel. This is actually a problem for which a linear function of the raw state can perfectly approximate the value function. Consider the case where $N = 1$. Then as an example, $Q(x, a = 2) = \theta_2' x$, if $\theta_2 = [-1, 1, -1, -1, -1]'$. If $N > 1$, then all elements of θ_a not corresponding to the rightmost column are 0.

The standard procedure in RL would be to learn θ_a for each $a \in \{1, ..., 5\}$, which provides an estimate of the Q-values. While the Q-values are in fact a linear function of the state, there are many distracting elements in the image, namely every black pixel not in the rightmost column, which impede the learning of θ. We compare our method to this linear function approximation method, where the coefficients are with least squares.

The first 200 time steps serve as a training period, in which random actions are selected and stored with the associated states and Q-values (rewards). At that point, we learn the feature mapping based on the Q-values. For the next 300 time steps, the raw image vector is mapped to the feature vector, and compared to all previous feature vectors. We compare the current image to the sets of previous images grouped by action, and select the action for which the Q-value of the nearest neighbor image is greatest. For the product kernel, the goal is to learn a weighted combination of Euclidean distances, resulting in a projection that only weights the pixels of interest in the right column. The dimensionality is effectively reduced to 5.

For the linear function approximation, at every time step after 200, the coefficients are learned by finding the least squares solution to $X_a\theta_a = Q_a$, for each action, where X_a is the matrix of the past states in which action a was used and Q_a is the vector of associated rewards. For each of the 300 testing images, the Q-value is estimated for each action, and the action with highest Q is selected.

We evaluate both methods based on the rate of correct choices, relative to all instances in which a black pixel is in the right column, out of the 300 testing images. The results are averaged over 20 Monte Carlo trials. We run the test for 7 values of N, and the results are seen in Figure 2. Our metric learning approach remains nearly perfect until $N = 100$, while the least squares approach declines with increasing state space size. By $N = 200$, which corresponds to a 1000 element state vector, with 6^{200} possible states, the performance of both methods has declined to only slightly better than random chance. The linear Q-learning approach fails for large state spaces because not enough data is available for it to capture the elements of the image which correlate with successful runs. This occurs because the method attempts to linearly combine all pixels in the image into the observed Q-values, which becomes an increasingly more difficult task as the image grows. Our method simply observes Q-values, and figures out a way to compare the images that produced similar ones.

4 Conclusion

The Q-values serve as a valuable indicator of the inherent similarity between states. Using a metric learning approach we can capture this similarity in a feature mapping. The resultant features are invariant to the distracting elements of the raw state, and learning proceeds much faster, as RL agents are able to generalize to previously unseen states when the essential task is the same. Our method is shown to perform extremely well in a problem where Q-values are given following each decision. Even for a space with 6^{100} states, only 200 time steps of interaction are required to achieve a 95% success rate, while linear Q-learning operating on the raw image is only 30% successful, despite the fact that the optimal solution is linear. Future work will apply our automatic feature generator alongside Q-value approximation techniques.

References

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

[2] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995.

[3] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. Littman, "An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning," in *25th Int. Conf on Machine Learning*, 2008.

[4] M. Petrik, "An analysis of laplacian methods for value function approximation in mdps," in *Proc. of the 20th Int. Joint Conf. on Artifical intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2574–2579.

[5] M. E. Taylor, B. Kulis, and F. Sha, "Metric learning for reinforcement learning agents," in *Int. Conf. on Autonomous Agents and Multiagent Systems*, 2011.

[6] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in Neural Information Processing Systems 15*, vol. 15, 2002, pp. 505–512.

[7] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[8] L. G. Sanchez Giraldo and J. C. Principe, "Information theoretic learning with infinitely divisible kernels," in *International Conference on Learning Representations*, May 2013.

Nexting and State Discovery in Robot Microworlds

Joseph Modayil, Adam White, A. Rupam Mahmood, Brendan Bennett, Darlinton C. P. Prauchner, Richard S. Sutton
Reinforcement Learning and Artificial Intelligence Laboratory
Department of Computer Science
University of Alberta, Canada

Abstract

We describe our recent work in reinforcement learning robots and its relationship to psychological ideas. We have recently shown how a robot can learn and make thousands of short-term predictions about its future stimuli, based on thousands of features, on-line and in real time. This is similar to the psychological phenomena of "nexting," in which animals learn to predict what sensory events will happen next, and sensory preconditioning. Our methodology is to study computational nexting in simple animal-like robots living in tightly controlled, small environments. This parallels a long tradition in artificial intelligence of studying "microworlds"—small simulated worlds, such as games and blocks worlds, that include important issues in a simplified form. Our use of robot microworlds is also analogous to the tightly controlled environments used when studying learning and brain function in the natural sciences. In ongoing and future work, we are exploring how nexting can provide a criteria for the discovery of state representations—memories or traces of past stimuli and actions that are helpful for making accurate predictions.

Keywords: Reinforcement Learning, Robotics, Temporal-difference Learning, Prediction, State Discovery

1 Nexting in Animals and Robots

Psychologists have noted that people and other animals seem to continually make large numbers of short-term predictions about their sensory input (Brogden 1939, Pezzulo 2008, Carlsson et al. 2000, Levitin 2006). People predict the sound made by their footsteps, the next note in a melody, or when an object will hit their eye. Making predictions of this simple, personal, short-term kind has been called *nexting* (Gilbert 2006). Nexting could be considered the core phenomenon studied in classical conditioning, which was described by Rescorla (1988) as

> "... the learning that results from exposure to relations among events in the environment. Such learning is a primary means by which the organism represents the structure of its world."

We have demonstrated a computational version of nexting on a small mobile robot, the *Critterbot* (Modayil, White & Sutton 2012). The Critterbot has a rich set of several dozen sensors. These were processed into a vector of six thousand cues, or features, which were then used to learn and make thousands of predictions ten times per second, in real time. These predictions provide the robot with a basic awareness of its environment, a substantially novel capability for robots.

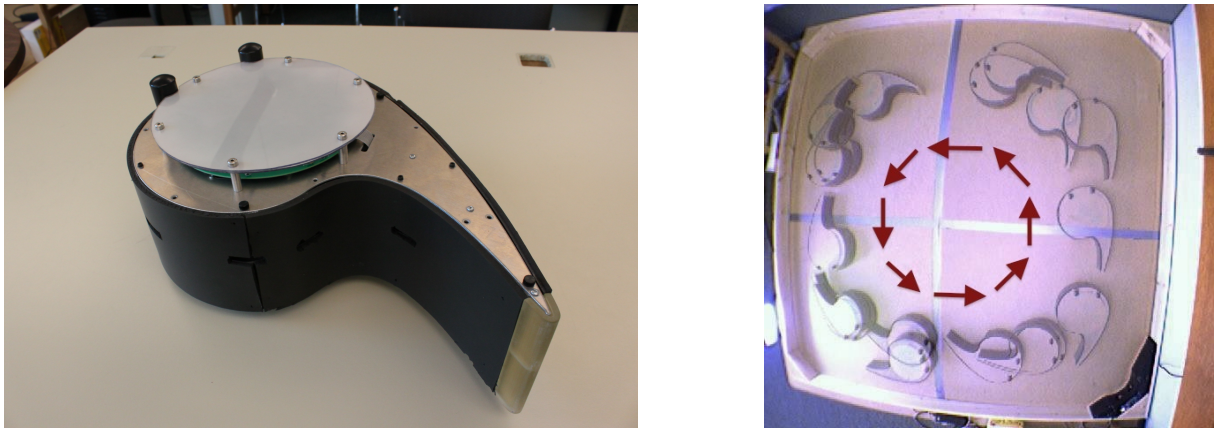


Figure 1: (left) The Critterbot is a sensor rich custom-designed mobile robot. (right) This robot moves around its pen by following the walls, and regularly passes a bright light on the lower left side.

In our experiment, the custom-built Critterbot robot circled in its pen, as shown in the right side of Figure 1. The robot learned in real time, making thousands of predictions about its sensory input signals at timescales from 0.1 to 8 seconds. The predictions were a generalization of the value functions commonly used in reinforcement learning, where an arbitrary function of the sensory input signals was used as a pseudo reward, and the discount rate determined the timescale. The learning was done by multiple copies of the TD(λ) temporal-difference learning algorithm, one for each prediction, operating in parallel.

As shown on the left of Figure 2, the robot anticipated when it would pass the light (shown on the lower left of the right panel of Figure 1). The same process was repeated for all the robot's sensors at several timescales, and the graph on the right of Figure 2 shows that the learning algorithm achieved substantial accuracy for most of the predictions within three hours.

2 A Create Microworld

The success of the nexting experiment shown above is surprising, because the predictions were learned as a function of the robot's current observation and previous action. The experiment shows that the Critterbot's sensory observations are rich enough to make accurate predictions using only the information available from a single timestep. However, robots need the ability to learn in environments with perceptual aliasing, and the Critterbot in its pen has too much sensory information to easily study this issue.

We have developed a *Create microworld* to study how an agent's state-representation, learning, and behaviour can interact when sensory information is constrained. For this microworld, we have chosen the Create robot, which is commercially available from iRobot. This is a robust yet inexpensive mobile robot adapted from an autonomous robot vacuum cleaner. The robot, shown in Figure 3, has two wheels and several sensors including four downward facing infrared sensors on its front that measure reflected infrared light from the ground. The robot operates on top of a dark disk set on a bright background, and low-level software routines use the infrared sensors to prevent the robot from leaving the disk. The boundary between the light and dark regions serves as a virtual wall, and the downward facing sensors serve as binary

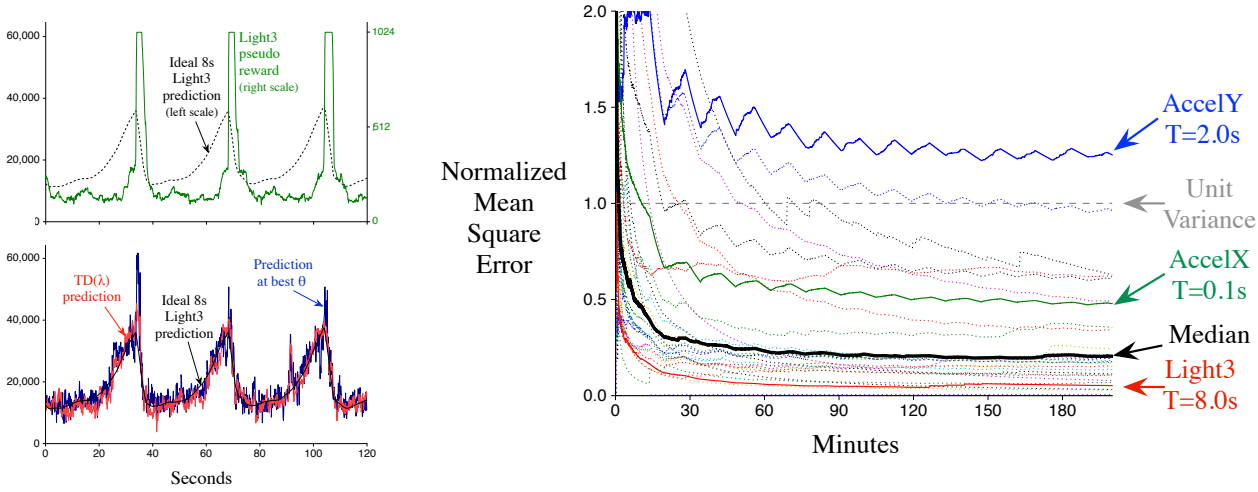
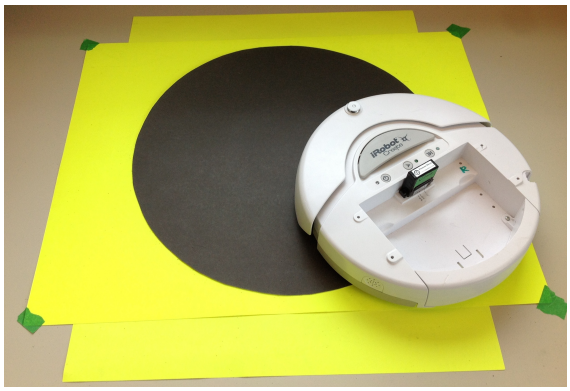


Figure 2: Nexting predictions. On the upper left, we show the signal from a light sensor on the robot, which rises sharply as the robot passes the bright light in the corner, along with the theoretically ideal prediction computed after the experiment, that anticipates the rise and fall of the sensor reading. In the lower left, we show the prediction learned and made in real-time by the TD(λ) algorithm, and the best prediction possible with the state-features used by the learning algorithm (computed offline). On the right we show learning curves for a selection of the sensory predictions.



	Sensor Observation				Wheel Action	
Time	Left	Front-Left	Front-Right	Right	Left	Right
1	0	0	0	0	+	+
2	0	0	0	0	+	+
3	0	0	0	1	-	+
4	0	0	1	1	+	-
5	0	1	1	1	+	-
6	0	0	0	1	+	-
7	0	0	0	1	+	-
9	0	0	0	1	+	-
10	0	0	0	1	+	-
11	0	0	0	1	+	-
12	0	0	0	1		

Figure 3: The Create microworld consists of a mobile robot constrained by its behaviour to explore a small space. Four downward facing sensors on the front of the robot detect the color difference on the paper sheets. This boundary serves as a virtual wall, and the sensors serve as whiskers. At each timestep, an observation vector indicates contact (1) or non-contact (0) with the "wall" and the action selects a direction of rotation for each wheel.

whiskers that sense this wall. The robot is thus operating in a setting analogous to an animal in a dark open room, as a single observation when the robot is not touching the wall provides no information to indicate whether the robot is near the wall or far away.

This environment is simplified from the previous example, but here we also find precedents in animal learning experiments, both in conceptualization and in experimental constraints. Our domain with one enclosed circular room is analogous to the Morris water maze, which is a circular testing environment for rats and mice that deliberately removes sensory information from the environment to study navigation. Constraints for wired and wireless communication are analogous for animals and robots. A wired interface to a desktop computer constrains free motion of the robot and the wires can get tangled, but wireless communication via Bluetooth can suffer from intermittent communication problems. The robot has enough power to operate for 3 hours without interruption, but human intervention is required after that point.

3 State Discovery

Psychologists have studied how animals with access to limited sensory information can use a state-representation make accurate predictions. Studies with trace conditioning have shown that animals can use a short-term memory of a past stimulus to anticipate when a future event will occur. It has also been noted that animals take longer to learn in such conditions than when relevant information is immediately available on the animals' sensors.

We are starting to study how a learning agent can incrementally discover a state-representation to perform well on a fixed set of prediction tasks. Our approach is to provide a fixed set of predictive nexting questions that serves as a measurable objective for evaluating the utility of state-features. Within the Create microworld, predictions about when the robot will sense a virtual wall will be inaccurate if the agent's state-representation is a function only of its current observation. We believe the learning agent can discover a better state-representation incrementally, by proposing new features and evaluating their utility for the fixed set of predictions (Mahmood and Sutton 2013).

References

Brogden, W. (1939). Sensory pre-conditioning. *Journal of Experimental Psychology 25*(4):323–332.

Carlsson, K., Petrovic, P., Skare, S., Petersson, K., Ingvar, M. (2000). Tickling expectations: neural processing in anticipation of a sensory stimulus. *Journal of Cognitive Neuroscience 12*(4):691–703.

Gilbert, D. (2006). *Stumbling on Happiness*. Knopf Press.

Levitin, D. (2006). *This is Your Brain on Music*. Dutton Books.

Mahmood, A. R., Sutton, R. S. (2013). *Representation Search through Generate and Test*. To appear in *Proceedings of the AAAI Workshop on Learning Rich Representations from Low-Level Sensors*, Bellevue, Washington, USA.

Modayil, J., White, A., Sutton, R. S. (2012). Multi-timescale Nexting in a Reinforcement Learning Robot In *Proceedings of the International Conference on Adaptive Behaviour*, pp 209–309.

Pezzulo, G. (2008). Coordinating with the future: The anticipatory nature of representation. *Minds and Machines 18*(2):179–225.

Rescorla, R. (1988). Pavlovian conditioning: It's not what you think it is. *American Psychologist*, 43(3):151–160.

Linking total movement history to action learning

Tom Stafford
Department of Psychology
University of Sheffield
Sheffield, England, S10 2TP
t.stafford@shef.ac.uk

Martin Thirkettle
The Open University
Walton Hall, Milton Keynes
United Kingdom, MK7 6AA
martin.thirkettle@open.ac.uk

Abstract

We have developed a new behavioural task which requires the agent to identify a target movement via exploratory movements. This conceptually simple, but versatile, paradigm allows the full history of movements made to be linked to learning of actions. Here we review recent results which show how the task can elucidate different aspects of the functional and anatomical basis of reinforcement learning in the human motor system. We also present a novel analysis which captures the relative influence of previous movements on learnt actions, over each point in the path of that learnt action. From a reinforcement learning perspective, this analysis can be thought of as revealing the shape of the eligibility trace: the relative strength of credit assignment to the motor efference copy across time.

Keywords: action-outcome learning, efference copy, eligibility trace

Acknowledgements

This research was funded by the European Community's Seventh Framework Programme FP7/2007-2013, 'Challenge 2 - Cognitive Systems, Interaction, Robotics', under grant agreement No. FP7-ICT-IP-231722, project "IM-CLeVeR - Intrinsically Motivated Cumulative Learning Versatile Robots" 2009-2013.

1 Introduction

1.1 Action acquisition rather than response frequency moderation

Historically, the main focus of behavioural research on learning has been on the moderation of response frequency. The initial acquisition of actions is a distinct topic (Redgrave & Gurney, 2006). In a review of the literature on operant conditioning Staddon & Niv (2008) note that it is a 'historical curiosity that almost all operant-conditioning research has been focused on the strengthening effect of reinforcement and almost none on the question of origins, where the behaviour comes from in the first place.'

Consideration of the computational framework for understanding operant conditioning, Reinforcement Learning, makes this point clear (Sutton & Barto, 1998; Woergoetter & Porr, 2007). Although reinforcement learning focussed on the optimal algorithm for updating the value of different actions according to sampling of their consequences, it requires that all possible actions be defined in advance (i.e. that the representation of the 'action space' is known).

A seminal example of the alternative focus on action acquisition is Thorndike's work (Thorndike, 1911), and his famous experiments looking at cats learning to escape from a box. Thorndike recorded only escape time, but through this variable, showed how initial exploration by the animal was refined over repeated attempts until the key components, and only those, could be rapidly selected by the animal to generate a predicted change on the world, which made possible the goal of escape. Thorndike's paradigm captured the outcome of the process of searching motor space and refining exploratory movements into learnt actions.

1.2 The short-latency dopamine signal in action acquisition

It is widely agreed that the subcortical basal ganglia play a key role in habit learning, reinforcement learning and action valuation. One focus has been the role of the short-latency dopamine signal. This has famously been associated with the reward prediction error, a key variable in reinforcement learning schultz:1997. Our position has been somewhat different. Due to timing constraints (Redgrave, Prescott, & Gurney, 1999), it has been proposed (Redgrave & Gurney, 2006) that the short-latency dopamine signal conveys a 'sensory prediction error', rather than reward prediction error. The difference being that rather than support action valuation, the main functional role of the short-latency dopamine signal, and the anatomical network within which it operates, is to allow the animal to identify from the recent history of movements those behaviours which trigger unexpected outcomes, to refine those behaviours and allow their storage in the repertoire of actions. The work presented here was inspired by this position. We have attempted to develop a new behavioural task which will provide insights into action-outcome learning, and specifically into the processes by which new actions are first identified and refined.

2 A novel task for investigating action acquisition

The essence of our task is to allow the agent free movement of a manipulandum. A subset of possible movements triggers a signal, which the agent seeks to learn to repeat. Previously we have used a joystick as the manipulandum, although it is conceivable to use a stylus (Shah, Thirkettle, Tidman, & Gurney, n.d.), touchscreen, or even, using motion tracking, free movements without a manipulandum (as do Wang et al., 2012, in a related task).

The target motion can be defined with respect to any movement parameters, but it is easiest to consider the cases of target locations or trajectories. The work we present here has used target defined by location, which we colloquially call 'the hotspot'. Typical single trial interaction with the task is shown in Figure 1. A full description of the task, including variants, and results showing basic learning phenomena has been published (Stafford et al., 2012).

3 Recent results

3.1 Evidence for subcortical dopaminergic involvement in action acquisition

Recently we have used precise stimulus control to affect the neural pathways down which reinforcing signals first travel (Thirkettle et al., 2013). Participants carried out the action acquisition task, as described above, with correct movements signalled by a stimulus which either changed in luminance or colour-shifted in a way that only the retinal S-cones are sensitive to. This allowed us to contrast action acquisition in which feedback has priority access to the retinotectal pathway (and so to the subcortical sites of the release of dopamine) with action acquisition in which feedback is restricted to a retinocortical route (and hence must trigger subcortical dopamine via one or more intermediate synapses). The results demonstrate that action acquisition is impaired when visual reinforcement signals must first access cortex, rather than having priority access to the subcortical visual system. This validates the claim that the superior colliculus is the primary trigger of short-latency dopamine in the basal ganglia. Previously we had formed this hypothesis, based on the

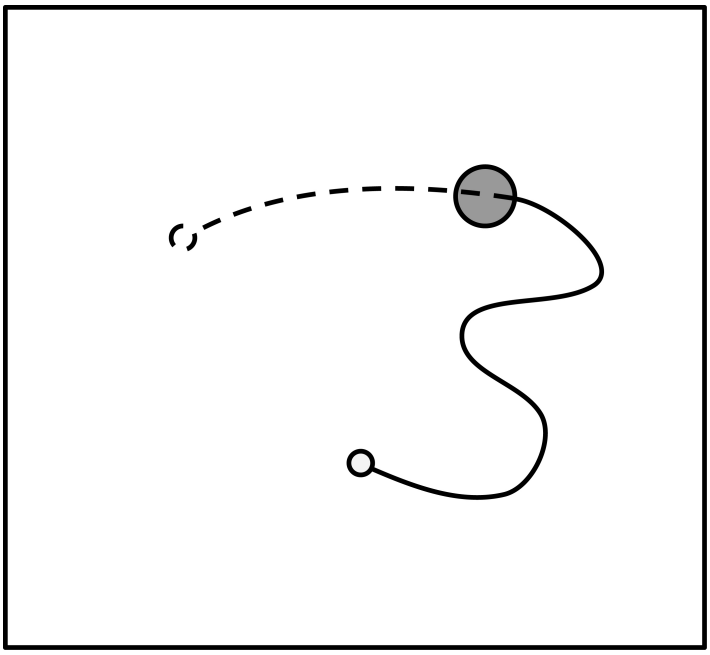


Figure 1: Schematic representation of a typical single trial. The agent's movement begins at the filled circle and from there they explore (movement shown by black line) until they encounter the target (in this case the location marked by a larger grey circle). Depending on their movement speed, alertness and the delay of the reinforcement signal, their movement will then proceed for some variable distance (dotted line). N.B. The task is typically carried out with no visual feedback, bar a signal when the agent's movement matches the target.

argument that the neural architecture of the basal ganglia is better suited to supporting action acquisition than action valuation via reward prediction errors (Redgrave & Gurney, 2006).

3.2 Our task is minutely sensitive to feedback delays

We have also shown that action acquisition in this task is minutely sensitive to signal delays, whether created experimentally or endogenously by sensory transmission times (Walton, Thirkettle, Gurney, Redgrave, & Stafford, 2013). This is consonant with our view that it is the short-latency dopaminergic signal which is instrumental in action acquisition. A key claim of Redgrave and Gurney (2006) is that the very short latency of the dopamine signal with respect to sensory feedback is due to a need to minimise contamination of the motor efference copy by irrelevant (i.e. post action) movements. The sensitivity of our task to delays is of the same range as the latency of the dopamine response.

3.3 Using the task to reveal priors for action acquisition

Comparing versions of the task with fixed and random starting points (Thirkettle et al., 2013) has allowed us to conclude that human action learning, in this task, is biased to assume trajectories rather than locations as the key unit of reinforcement. The fixed start point conditions allowed successful target learning, whereas the random starting position conditions did not (which we presume is because the fixed start position conditions allowed a location-defined target to be achieved by a simple trajectory, which the random starting position conditions did not). This result matches another recent result by Dam and colleagues (Dam, Kording, & Wei, 2013), who independently from us used a similar task in the context of reinforcement learning and movement control. Like us, they showed that human participants, using exploration-guided movements, were able to reduce error in an action-space defined by both direction and shape of movement (but that shape of movement, analogous to our trajectory-defined targets, dominated learning).

3.4 Demonstrating the value of exploration in action acquisition

The exploration-exploitation trade-off is a well known feature of reinforcement learning systems (Sutton & Barto, 1998). Analysis of performance on our task, across trials, shows that high early variability is associated with improved subsequent performance (Stafford et al., 2012) — exactly what would be expected if the motor system of participants was able to use, or induce, variability in movement in order to better explore the space of movements which best meet the target criterion. This results holds for humans and rats on our task, and has recently been confirmed in analysis of motor skill learning in a completely different task (Stafford & Dewar, 2013).

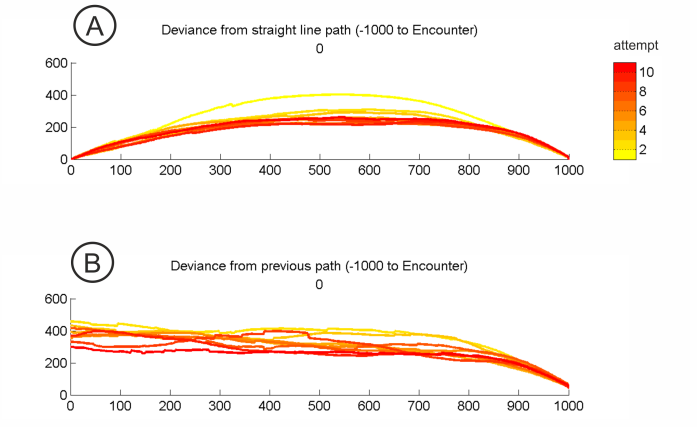


Figure 2: The difference between successive movement paths and the straight-line path (A) The difference between successive movement paths and the preceding path (B) for attempts 2 - 10 at learning a target, averaged over 15 participants and 3 targets.

4 Revealing the temporal window of credit assignment in motor control

A great advantage of this task is it allows the total record of movements made to be linked to the actions that are learnt. Over multiple attempts participants refine their movements, under feedback indicating when they successfully match the target location. For each of these attempts we can trace the recent movement history — defined as the positions of the manipulandum in 1000 ms before achieving the target (1000 data points, since we poll at 1000 Hz).

Analysis of data collected from the task shows that successive attempts at locating the target movement produce paths which are closer to a direct line between the origin and the target - i.e. the path that minimises movement length (Figure 2A). This effect is most pronounced in the middle of the range of point analysed. At the start and end of the movement paths the movements are highly constrained and so differ little from the direct line path. This suggests that successive movements become increasingly optimal, is so much as a minimal distance can define optimal. A second analysis is to compare successive paths against each other, rather than against the direct line path. This analysis, Figure 2B, shows that the difference between successive movements diminished. This effect is most pronounced in in the middle range of point analyses, again the definition of the end of the movement as reaching the target constrains how different points near the end of the movement can be. This suggests that successive movements become increasingly stereotyped.

Together these analyses suggest that while the history of movements made in preceding trials exerts some influence on the learnt action, that the learn action is also progressively refined to become more efficient (as benchmarked by a a straight-line towards the target).

Considering these two forces, 'optimisation' and 'habit', we can compare their relative influence at each point along the movement path by calculating a ratio of the values for these two plots (i.e. the ratio of distance from the preceding path to the distance from the optimal path). This indicates, at each point along the path towards the target, the extent to which the movement is based on previous movements rather than on the 'optimal' (straight line) movement. The results of this analysis are shown in Figure 3.

One interpretation of the pattern revealed is as reflecting the eligibility trace. One solution to the credit assignment problem in motor control is to use a continuous function which describes the diminishing weight that actions in the motor history are accorded in associating actions with effects (Barto, Sutton, & Brouwer, 1981; Singh & Sutton, 1996). Typically this function weights recent actions disproportionately more than actions which are more distant in time. In our case, this would mean that aspects of the movement which are closest in time to reinforcement will be most eligible. A side-effect is that incidental movements made near the target, even though they are not causal, are most likely to be incorporated into the learn action (and conversely least likely to be removed in the process of refining the action). This is what we see in Figure 3.

Future analysis will look at how the shape of this ratio measure matches proposals for the eligibility trace in machine learning, at how the measure is affected by artificially induced reinforcement delays and whether there are reliable individual differences on the measure and if these can be related to the efficiency of action learning.

References

Barto, A. G., Sutton, R. S., & Brouwer, P. S. (1981). Associative search network: A reinforcement learning associative memory. *Biological cybernetics*, *40*(3), 201–211.

Dam, G., Kording, K., & Wei, K. (2013). Credit assignment during movement reinforcement learning. *PloS one*, *8*(2), e55352.

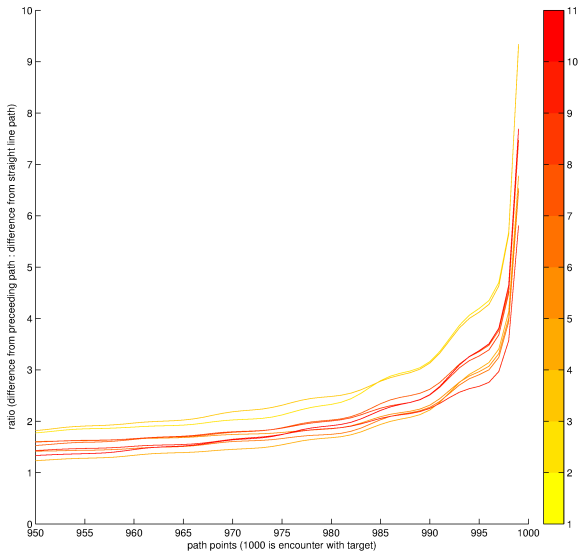


Figure 3: The shape of the eligibility trace in motor learning? The ratio of the difference between each attempt path and the preceding path to the difference between that attempt path and the straight-line path. For attempts 2 - 10 at learning a target, averaged over 15 participants and 3 targets. (same data as Figure 2)

Redgrave, P., & Gurney, K. (2006). The short-latency dopamine signal: a role in discovering novel actions? *Nature Reviews Neuroscience*, *7*(12), 967–975.

Redgrave, P., Prescott, T., & Gurney, K. (1999). Is the short latency dopamine response too short to signal reward error ? *Trends Neurosci.*, *22*, 146-151.

Shah, A., Thirkettle, M., Tidman, J., & Gurney, K. N. (n.d.). Emergent variation and repetition trade-off in a biologically-plausible model of intrinsically-motivated action discovery. *under review*.

Singh, S., & Sutton, R. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, *22*(1-3), 123–158.

Staddon, J., & Niv, Y. (2008). Operant conditioning. *Scholarpedia*, *3*(9), 2318.

Stafford, T., & Dewar, M. (2013). Testing theories of skill learning using a very large sample of online game players. In *Proceedings of the 35th cognitive science society annual conference.*

Stafford, T., Thirkettle, M., Walton, T., Vautrelle, N., Hetherington, L., Port, M., et al. (2012). A novel task for the investigation of action acquisition. *PloS one*, *7*(6), e37749.

Sutton, R., & Barto, A. (1998). *Reinforcement learning - an introduction.* Cambridge, MA: MIT Press.

Thirkettle, M., Walton, T., Shah, A., Gurney, K., Redgrave, P., & Stafford, T. (2013). The path to learning: Action acquisition is impaired when visual reinforcement signals must first access cortex. *Behavioural brain research*, *243*, 267–272.

Thorndike, E. (1911). *Animal intelligence.* New York: Macmillan.

Walton, T., Thirkettle, M., Gurney, K., Redgrave, P., & Stafford, T. (2013). The discovery of novel actions is affected by very brief reinforcement delays and reinforcement modality. *Journal of Motor Behavior*, *45*(4), 351–360.

Wang, Q., Bolhuis, J., Rothkopf, C. A., Kolling, T., Knopf, M., & Triesch, J. (2012). Infants in control: Rapid anticipation of action outcomes in a gaze-contingent paradigm. *PloS one*, *7*(2), e30884.

Woergoetter, F., & Porr, B. (2007). Reinforcement learning. *Scholarpedia*, *3*(3), 1448.

Cue Competition in Human Incidental Learning

I.P.L. McLaren[1]
School of Psychology
University of Exeter
i.p.l.mclaren@exeter.ac.uk

F.W. Jones
School of Psychology
Canterbury Christ Church University
f.w.jones@exeter.ac.uk

R.P. McLaren
School of Psychology
University of Exeter
r.p.mclaren@exeter.ac.uk

F. Yeates
School of Psychology
University of Exeter
fy212@exeter.ac.uk

Abstract

There is a question as to whether cue competition effects can be observed in incidental learning paradigms in humans. Some authors have reported that cue competition is not observed, suggesting that previous demonstrations of cue competition have relied on explicit awareness of the task in hand. This would in turn imply that these effects are more likely to be the product of cognitive inference than associative learning. We addressed this question by using two paradigms previously shown to produce associative learning under incidental conditions. One was a standard SRT task in which the preceding two trials of a run of three predicted the third 2/3 of the time, and the other was based on another predictive cue, a colored square, which could also stochastically predict the next response required. Both tasks were run under incidental conditions, and we have demonstrated in other studies that both cues would support learning in these circumstances in the absence of any verbalisable knowledge of the rules involved. The question was to what extent would these two cues compete if run concurrently, as assayed by their ability to make the next response faster and more accurate than controls? We assessed this by comparing a dual cue group to a color only control and a sequence only control. Our results showed that all three groups learned, but that during a test phase where each cue could be assessed independently, the dual group showed a marked decline in performance relative to the color control, and very similar performance to the sequence control. We interpret this as evidence for overshadowing occurring between the two predictive cues in the dual group, such that when combined their performance would be equivalent or superior to either control, but when assessed independently, the color cue actually has a weaker association to the outcome than the equivalent cue in the control group. We conclude that the sequence cues overshadowed the color cues in this task, and discuss possible theoretical accounts of this phenomenon.

Keywords: Cue competition, Overshadowing, Associative, Incidental

Acknowledgements

This research was supported by an ESRC grant to IPL McLaren and FW Jones.

[1] http://psychology.exeter.ac.uk/staff/index.php?web_id=Ian_McLaren

Cue Competition in Human Incidental Learning

We start by considering the phenomenon of overshadowing, as this is a paradigmatic example of cue competition, a domain that also encompasses blocking (Kamin, 1968). The result here is that, if two quite distinct, equally salient cues, A and B, are trained in compound to predict a US, then responding to either A or B is less than would be seen if that cue had been trained in isolation. If one cue, say A, is more salient than B, then it tends to dominate learning, and relatively little accrues to B (see Mackintosh, 1976, for just such an experiment). This result is easily explained by associative theories. According to the Rescorla-Wagner model, the two cues, A and B, share the associative strength to the outcome between them in proportion to their relative salience. Pearce's (1987) configural theory arrives at the same result by a different route, arguing that learning about AB generalises only imperfectly to A or B, again to an extent determined by the relative salience of the cues involved. Associative theories, then, provide good explanations of cue competition phenomena.

Equally, however, there is no doubt that an approach based on cognitive inference can explain overshadowing, by taking the view that the subjects in the experiment are using a heuristic of the type "if there are two cues predicting the outcome, then credit for this prediction must be shared between them according to their salience". For example, this heuristic can be used to explain the results of allergy prediction paradigms such as in Le Pelley and McLaren (2001), where a combination of two foods, A and B, predict an allergic reaction in a hypothetical patient, "Mr. X". The result is that the ratings for A and B are less than that for control cues trained on their own to predict the same outcome.

If the results of such experiments are equally well explained by either associative or inferential accounts, how will it be possible to decide between them? In humans, one way may be to use procedures that make it unlikely that participants will be able to employ cognitive inference – which we assume relies on working memory which has a limited capacity. Le Pelley, Oakeshott and McLaren (2005) argued that using many different trials, presented in a random order, each employing some of a large number of stimuli with different relationships to the available outcomes should make it hard for participants to keep explicit track of the contingent relationships in the experiment. Le Pelley and McLaren (2001) were also at pains to use these conditions (high memory load due to using many cues and trial by trial presentation) for similar reasons, so it seems plausible to argue that the cue competition effects they observed were probably associative in origin, but we cannot be certain that this was the case. In many other cases, where few cues are used and memory load is low, the rating given may well owe more to cognitive inference than associative learning.

One particular version of this inferential heuristic for overshadowing requires that the subject, whether animal or human, knows which cues predict which outcomes, and then uses this information to generate behaviour. We can characterise this account of overshadowing as reliant on explicit memory as well as learning. This explanation of overshadowing takes on particular relevance when we consider the claim that humans do not show cue competition effects (Jimenez and Vazquez, 2011) under incidental conditions. If this is because people do not have access to the necessary explicit cue-outcome information required for cognitive inference to be brought to bear and hence produce overshadowing, then this would be good evidence that humans learn propositionally, and when the relevant information is unavailable, cue competition effects do not occur. It would also suggest that reinforcement learning in humans is driven by different mechanisms to those in other animals. If, on the other hand, cue competition effects could be demonstrated under incidental conditions in humans in the absence of explicit cognitive inference, then this would be entirely consistent with an associative account of learning under these conditions, and would suggest strong parallels between human and infra-human learning in these circumstances.

Design issues

We have already indicated that demonstrations of overshadowing using the allergy prediction paradigm, whilst robust, are susceptible to the complaint that they may be propositionally driven rather than associatively mediated. A second issue is that the stimuli that serve as the CSs in these experiments may be too similar in kind, in that they are both foods. The analogy would be to an animal experiment in which the overshadowing was demonstrated to two tones of different pitch, rather than a tone and a light. The former might give rise to concerns that the two tones when played together interacted in some way so as to change their stimulus quality, and that this interaction was lost when presented individually, so that the reduction in rating that occurred on test could be explained by some change in the perceived stimulus. No such

process would apply when the stimuli were trained alone. It would clearly be better if the two CSs were different in kind so that this type of potential confound could be avoided. Our two classes of cue were chosen to have quite distinct characteristics to avoid this problem. We employed a basic SRT paradigm similar to that of Willingham, Nissen and Bullemer (1989), in which there were two circles that defined two stimulus locations, left and right. At the start of a trial, the circles are outlines, then one of them fills, and the corresponding key has to be pressed. Unknown to the participants, in those groups that were given sequential information, there was a 2/3 chance of a trial being predicted by the two preceding trials. The rule was that if the two preceding trials were both the same, then that trial was likely to be an X, whereas if they were different, it was likely to be a Y, with the response assignments for X and Y counterbalanced across participants. Thus, the first type of cue was provided by the sequence of locations that occurred / responses required. The second cue type was provided by a colored square that flashed up before the circle filled in, presented at fixation between the two circles. Participants for whom color information was relevant had a 3/4 chance that the color would predict the response location on half the trials. On the other half of trials different colors were used that were not predictive, and these could be used as color control trials. We settled on these parameters for the tasks on the basis of extensive piloting and prior work, to ensure that both the sequential information and the color information were capable of supporting learning under incidental conditions, but without participants inducing the rule relating either type of cue to the required response (see Jones and McLaren, 2009 for more on the sequences, and Yeates, Jones, Wills, Aitken and McLaren, 2012, 2013 for details on the colour task). Table 1 gives the stimulus construction for each group.

GROUP	S1	S2	S3	S4	S5	S6	S7	S8
COLOR	XXX 431	XXY 242	XYX 321	XYY 332	YXX 141	YXY 412	YYX 431	YYY 342
SEQUENCE	XXX 431	XXX 242	XYY 321	XYY 332	YXY 141	YXY 412	YYX 431	YYX 342
DUAL	XXX 431	XXX 241	XYY 322	XYY 332	YXY 142	YXY 412	YYX 431	YYX 341

Table 1: This shows the construction of the stimulus sequences and contingencies for the three groups (N=30 for each group) in an idealised form to convey the relationship between the groups. Sequences and mappings were randomised / counterbalanced where appropriate. The letters (X, Y) stand for left/right responses, and the numbers (1, 2, 3, 4) for colors. Stimuli shown in red are those changed with respect to Group Color. In Group Color colors 1 and 2 are predictive, and 3 and 4 act as controls. All 8 sequence triplets are shown that were used to construct the pseudorandom trial order. In Group Sequence no color is predictive, but only 4 sequence triplets are used so that e.g., XX is typically followed by X. In Group Dual colors 1 and 2 are once again predictive, and so are the sequences. There were 16 blocks of training and 2 blocks of test, with each block containing 96 trials.

Another point worth raising is that in all the experiments (that we are aware of) that have studied overshadowing in humans that come close to meeting our first two conditions, i.e. incidental learning with dissimilar cue types, the comparison has been between CSs trained in compound and tested individually, and a group or groups trained with the individual CSs and then tested. The problem with this procedure is that one group experiences a major change from training to test (the compound group) whereas the other does not. This, on its own, may be enough to depress responding in the compound group if they come to believe that circumstances have changed and deliberately and strategically alter their responses as a consequence (something that seems intuitively less likely to be the case in a rat or a pigeon). Note that this is not the same as a generalization decrement account of overshadowing that would, for example, follow from Pearce's (1987) configural model. It is rather an appeal to a strategic decision based on changing circumstances during the course of the experiment, and we avoided this in our design by making sure that the transition from training to test was unsignalled and unlikely to be noticeable. Table 1 shows how Group Dual had both sequence and color information programmed in, Group Color had the same type of color information as Group Dual, and Group Sequence had the same type of sequence information as Group Dual. Group Sequence were still shown a colored square just before the response location was indicated, but the color bore no relation to that location; equally Group Color experienced sequences of trials in just the

same way as Group Dual, but these were not predictive. The point is that all groups experienced a fast-paced sequence of trials (mean RT = 300 msec, and mean errors per block = 4) cued by a colored square during both training and test, so there should be no obvious difference in their subjective experience, and no obvious difference between training (when the contingencies were as shown in Table 1) and test (when all contingencies were 50:50, and sequence and color information were uncorrelated).

Results

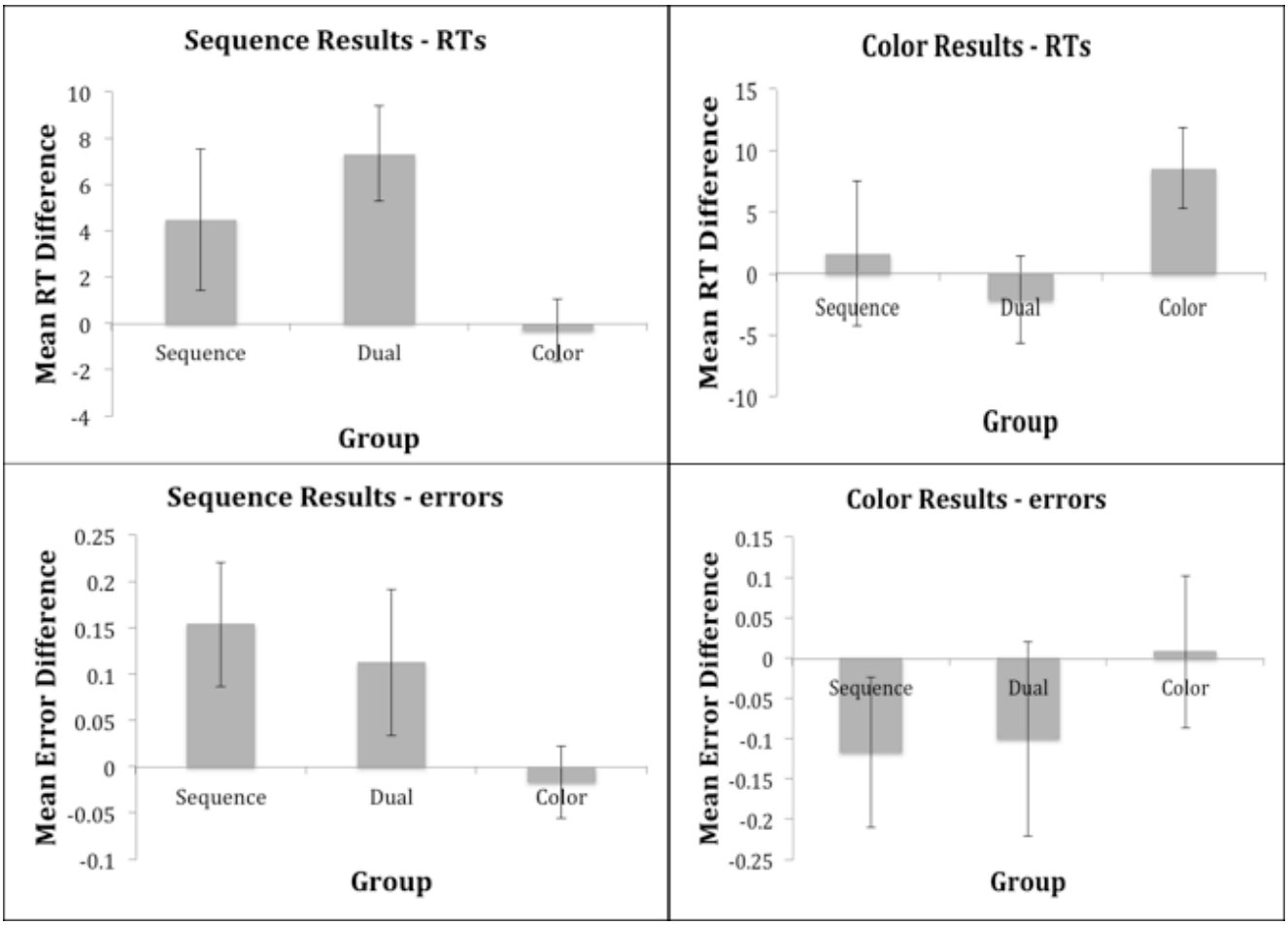


Figure 1: The top panels display the mean differences in RT (msec) and their standard error analysed by sequence (left panel: untrained sequences–trained sequences), and by color (right panel: untrained colors–trained colors). The corresponding mean error differences and their standard errors are shown underneath.

Figure 1 shows the test results for sequence and color learning. The data shown in Figure 1 are the mean difference between trained and untrained sequences (left panels) or colors (right panels) in RTs (top panels) and errors (bottom panels) for each of the three groups. Higher scores indicate more learning (chance is zero), and it is clear that both the Sequence and Dual groups showed good evidence of sequence learning (left panels) as measured by RTs and errors, whereas there was little evidence of learning in the Color group on this measure. Given that the test phase was, in effect, an extinction treatment, the evidence for sustained performance on the basis of what had been learned during training in Group Sequence and Group Dual is noteworthy and implies strong learning of the sequence information available during training.

The right hand panels show the difference scores obtained by comparing performance for the predictive colors with the control colors for RTs and errors on test. Group Dual showed no evidence of learning about the colors on either the RT or the error measure (unsurprisingly, the same was true for Group Sequence). Although Group Color also showed little or no evidence of learning on the error measure, their RT performance showed a significant effect, and was significantly better than that of the other two groups.

Conclusions

It would appear, then, that the Dual group learned about the sequences, but did not learn the color

information available to them, even though Group Color shows that this was eminently possible. This is what would be expected if the sequence cues had overshadowed the color cues in the Dual group (but not vice-versa). Thus, it is possible to demonstrate cue competition effects in humans trained under incidental conditions, and we believe this to be the first such demonstration. Were the participants in this experiment aware of the sequence or color rules? Previous experiments and pilot work suggest that this should not be the case, and post-experiment interviews established that participants were unable to give any accurate information about the sequences, or say which colors were predictive. Crucially, there was no difference between Dual and Color groups in terms of their ability to guess which colors were predictive (44% and 45% respectively), and both values are numerically below chance (50%). Equally, there was no reliable difference in the proportion asserting that the sequences were random (using a conservative criterion in scoring this) in any of the Color, Dual or Sequence groups (50%, 57% and 70%), and the trend favoured the Sequence group as thinking that their sequences were random. Given these results, we can now reject the argument that cue competition in humans is only observed under intentional learning conditions, and its corollary that this is because it relies on explicit cognitive inference to manifest.

How might we explain this effect? As mentioned earlier, there are two standard accounts of overshadowing available, one based on competition for associative strength and exemplified by the Rescorla-Wagner (1972) model, the other based on the idea of generalization decrement with Pearce's (1987) configural model as its flag bearer. Neither are capable of learning the sequential information presented in this experiment, but error-correcting recursive networks such as the SRN (Elman, 1990) and its variants the RASRN (Yeates, Jones, Wills, McLaren and McLaren, 2013) and the APECS SRN (Jones, Le Pelley and McLaren, 2002; and see McLaren, Forrest and McLaren, 2012)) can, and contain one or both mechanisms for overshadowing. Thus, they are able to predict the results obtained here. We are also in a position to say that cue competition experiments in humans under incidental conditions produce the same type of result as that observed in other animals, which is consistent with there being some common basic mechanism for associative learning.

References

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14,* 179-211.
Jiménez, L., & Vázquez, G. A. (2011). Implicit sequence learning and contextual cueing do not compete for central cognitive resources. *Journal of Experimental Psychology:Human Perception and Performance*, 37, 222–235.
Jones, F.W., Le Pelley, M. E., & McLaren, I. P. L. (2002). The APECS-SRN: Towards a model of SRT sequence learning. *Proceedings of the World Congress on Computational Intelligence*.
Jones, F.W. and McLaren, I.P.L. (2009). Human Sequence Learning Under Incidental and Intentional Conditions. *Journal of Experimental Psychology: Animal Behavior Processes*, 35, 538-553.
Kamin, L.J. (1968). 'Attention-like' processes in classical conditioning. In M.R. Jones (Ed.) *Miami symposium on the prediction of behaviour: Aversive stimulation*. Univ. of Miami Press. pp. 9-33.
Le Pelley, M. E., & McLaren, I. P. L. (2001). Retrospective revaluation in humans: Learning or memory? *Quarterly Journal of Experimental Psychology*, 54B, 311-352.
Le Pelley, M. E., Oakeshott, S. M. and McLaren, I. P. L. (2005). Blocking and unblocking in humans. *Journal of Experimental Psychology: Animal Behavior Processes*, 31.
Mackintosh, N.J. (1976). Overshadowing and stimulus intensity. *Animal Learning and Behavior*, 4, 186-92.
McLaren, I. P. L., Forrest, C.L., and McLaren, R.P. (2012). Elemental representation and configural mappings: Combining elemental and configural theories of associative learning. *Learning & Behavior*, 320-33.
Pearce, J. M. (1987). A model of stimulus generalization for Pavlovian conditioning. *Psych. Review*, 94, 61-73.
Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and non- reinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical conditioning II: Current research and theory* (pp. 64-99). New York: Appleton-Century-Crofts.
Willingham D.B., Nissen M.J. and Bullemer P. (1989). On the development of procedural knowledge. *Journal of Experimental Psychology: Learning, Memory and Cognition, 15,* 1047-1060.
Yeates, F., Jones, F.W., Wills, A.J., Aitken, M.R.F. and McLaren, I.P.L. (2012). Implicit Learning: A Demonstration and a Novel SRT Paradigm. In: *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, Sapporo, Japan.
Yeates, F., Jones, F.W., Wills, A.J., McLaren, R.P. and McLaren, I.P.L. (2013). Modelling human sequence learning under incidental conditions. *Journal of Experimental Psychology: Animal Behavior Processes*.
Yeates, F., Jones, F.W., Wills, A.J., Aitken, M.R.F. and McLaren, I.P.L. (2013). Implicit Learning: A Demonstration and a Revision of a Novel SRT Paradigm. In: *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, Berlin, Germany.

A stochastic control mechanism for planning of goal directed behavior

H.J. Kappen[*]
Department of Neurophysics
Donders Institute for Brain, Cognition and Behaviour
Radboud University
Nijmegen, the Netherlands
b.kappen@science.ru.nl

J. Bierkens[†]
Department of Neurophysics
Donders Institute for Brain, Cognition and Behaviour
Radboud University
Nijmegen, the Netherlands
j.bierkens@science.ru.nl

Abstract

Navigation requires planning to previously remembered goal locations. In this paper, we propose KL learning which is an on-line version of KL control theory as a possible abstract mechanism to account for recent findings that show that sequences of place cell activity in rats strongly correlate with the animals future trajectory to remembered targets [1]. We show the convergence of KL learning for a restricted setting. We argue that KL learning is simpler than reinforcement learning and discuss possible neural implementation of KL learning.

1 Introduction

Effective navigation requires planning to goal locations that have been previously visited. The hippocampus has long been associated with navigation [2]. Hippocampal place cells fire selectively when an animal occupies a restricted location in an environment and are considered the neural substrate of an internal cognitive map of the environment that is necessary for flexible navigation, map-based spatial learning, and episodic memory.

Recently, [3, 1] have shown that before goal directed behavior, the rat hippocampus generates brief sequences encoding spatial trajectories from the current location of the animal to a goal location. These trajectories predict immediate future navigational behaviour to remembered targets [1].

It has been a challenge to obtain a computational mechanism to understand how individual place responses tied to the current location might be informative about other locations that the animal cares about, such as the remembered goal location. [4] propose that place cells can be modelled as the input to an actor-critic model of navigational learning, using the temporal difference learning rule.

Here, we propose an alternative explanation for learning goal-directed behavior based on KL (Kullback-Leibler) control theory as developed by [5, 6]. We assume the ergodic setting, also referred to as the average cost infinite horizon problem, where actions and rewards do not explicitly depend on time. We assume discrete states and discrete time. In this case the optimal control can be expressed in terms of an extreme eigenvector and eigenvalue [5]. We show that the optimal controls can be computed by sampling and we refer to this method as KL learning. This allows for an adaptive solution while the animal explores the environment that automatically updates its representation to changing rewards.

2 KL control theory

Goal directed behavior is naturally formulated as an optimal control problem, where negative cost is associated with the goal state and positive cost with an individual action. The solution to the optimal control problem is a sequence of actions that reaches the goal with minimal cost.

[5, 6] introduced the class of so-called KL control problems. We first discuss the finite horizon formulation. Let x label the states and $q(x'|x)$ denote a first order Markov process on this state space. We refer to $q(x'|x)$ as the *uncontrolled dynamics*. Consider a control problem to find a first order Markov process $p(x'|x)$ that minimizes

$$C = KL(p||q) + \langle V \rangle_p \,, \tag{1}$$

with $KL(p||q) = \sum_\tau p(\tau|x_0) \log \frac{p(\tau|x_0)}{q(\tau|x_0)}$ the Kullback-Leibler divergence between distributions $p(\tau|x_0)$ and $q(\tau|x_0)$ over trajectories $\tau = x_{1:T}$ according to the Markov processes $p(x'|x)$ and $q(x'|x)$, respectively and $\tau = x_{0:T}$ a trajectory of length T starting at a

[*] www.snn.ru.nl/~bertk
[†] jbierkens.nl

given initial state x_0. $\langle V \rangle_p = \sum_\tau p(\tau|x_0)V(\tau)$ is the expected cost under the optimal control distribution p and $V(\tau) = \sum_{t=1}^T V(x_t)$ is the cost of trajectory τ with $V(x_t)$ the immediate cost. It can be easily shown that the optimal control distribution that minimizes Eq. 1 and the optimal cost are given by

$$p(\tau|x_0) \quad = \quad \frac{1}{\psi(x_0)} q(\tau|x_0) \exp(-V(\tau)) \qquad \psi(x_0) = \sum_\tau q(\tau|x_0) \exp(-V(\tau))$$

and the optimal cost is $C(x_0) = -\log \psi(x_0)$. Since $q(\tau|x_0)$ is a first order Markov process, this shows that the optimal control solution $p(\tau|x_0)$ is also a first order Markov process. The state transition matrix of the optimal control solution is given by

$$p_t(x_t|x_{t-1}) \quad = \quad q(x_t|x_{t-1}) \exp(-V(x_t)) \frac{\beta_t(x_t)}{\beta_{t-1}(x_{t-1})} \tag{2}$$

where the functions $\beta_t(x)$ are found by message passing

$$\beta_{t-1}(x_{t-1}) \quad = \quad \sum_{x_t} q(x_t|x_{t-1}) \exp(-V(x_t)) \beta_t(x_t) \tag{3}$$

with $\beta_T(x_T) = 1$. Eq. 2 provides the optimal control solution in terms of a distribution over the states x_t given state x_{t-1}. Actions are generated by sampling from $p(x_t|x_{t-1})$.

2.1 Ergodic KL control

In the limit $T \to \infty$ the control solution becomes independent of time, as we show now. First, note that Eq. 3 is an instance of the power method $\beta_k = H\beta_{k-1}$ with $H(x, y) = q(y|x) \exp(-V(y))$ (with iteration number $k = -t$). When $t \to -\infty$ and when $q(y|x)$ is ergodic the solution converges to the unique extreme eigenvector of H, up to a (possibly infinite) multiplicative factor:

$$H\beta = \lambda\beta(x) \qquad H(x, y) = q(y|x) \exp(-V(y)) \tag{4}$$

with β, λ the Perron-Frobenius right eigenvector and eigenvalue of H [5]. Thus, the optimal control solution Eq. 2 becomes time-independent The ergodic process $q(y|x)$ has a unique stationary distribution (invariant measure) $q(x)$ that satisfies $q(y) = \sum_x q(y|x)q(x)$. When $q(y|x)$ satisfies detailed balance , so does $p(y|x)$ and its equilibrium distribution is

$$p(x) = \frac{1}{Z} q(x) \exp(-V(x)) \beta^2(x) \tag{5}$$

In the language of reinforcement learning, $\log \beta(x)$ can be viewed as to represent the optimal value of state x.

2.2 KL Learning

Instead of computing the extreme eigenvector of H using a linear algebra package, we can compute the solution incrementally by sampling the state space using the uncontrolled dynamics $q(y|x)$. Let $\mathbb{R}^n_+$ denote the set of $x \in \mathbb{R}^n$ such that $x(i) > 0$ for all $i = 1, \ldots, n$. Let $\beta_0 \in \mathbb{R}^n_+$ and $\lambda_0 = ||\beta_0||_1 = \mathbb{1}^T \beta_0$, where $\mathbb{1} = [1, 1, \ldots, 1]^T$ of length n. Consider the following update rules for x, β and λ.

$$\begin{aligned} x_t &\leftarrow \text{random sample from } q(\cdot|x_{t-1}) \\ \Delta &\leftarrow \frac{\exp(-V(x_t))\beta(x_t)}{\lambda} - \beta(x_{t-1}) \\ \beta_t &\leftarrow \beta_{t-1} \\ \beta_t(x_{t-1}) &\leftarrow \beta_{t-1}(x_{t-1}) + \eta\Delta \\ \lambda_t &\leftarrow \lambda_{t-1} + \eta\Delta \end{aligned} \tag{6}$$

It is straightforward to check that for all t, $\lambda_t = ||\beta_t||_1$. When $q(\cdot|\cdot)$ is ergodic and its stationary distribution is state independent, it can be shown that the stochastic trajectories of (6) converge in probability to $(\beta^\star, \lambda^\star)$.

3 Hippocampal model

We apply the KL control method to model the experimental setup of [1]. In their experiment, 4 well-trained rats were implanted with 40 tetrode electrode while performing a spatial memory task in a 2×2 meter open area. In each animal, they record up to 250 hippocampal neurons with well-defined place fields. Each trial consisted of two phases: in phase one, the rat was required to forage to obtain reward in an unknown location (Random). Directly afterwards, (phase two) the rat would obtain reward in a predictable reward location (Home). Latencies of trajectories to Random locations were significantly longer than to the Home location, indicating that the animal could remember the Home location but not the Random location.

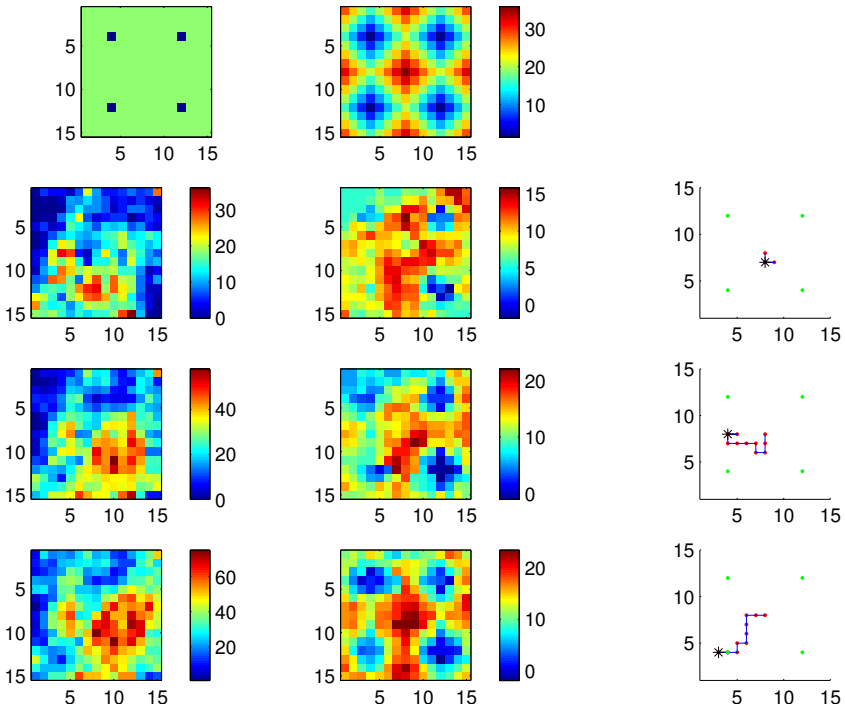


Figure 1: Model of 15×15 hippocampal place cells in a 15×15 grid world. Uncontrolled dynamics $q(x'|x) = 0.2$ when x, x' are the same or nearest neighbouring states (north east south west). Rat moves randomly according to q for $T = 8000$ iterations, $\eta = 0.5$. Top row: Left: Grid world with 4 food locations present. Middle: Exact solution $-\log \beta_{ex}(x)$ as the solution of the eigenvector equation $H\beta_{ex} = \lambda\beta_{ex}$. Second-fourth row: results after $t = 800, 4000, 8000$ iterations. Left: Histogram of states explored by the rat. Middle: $-\log \beta(x)$. Right: An exploration trial by the rat starting at the home location (8,8) for 12 steps using the controlled solution p. End location of the trial is marked by a $*$. Food locations are marked by green dots.

Candidate neural events were identified as brief increases in population spiking activity during periods of immobility while the rat performed the task. During many candidate events, decoded position revealed temporally compressed, two-dimensional trajectories across the environment with path length ranging from 40.0 cm to 199.1 cm (see [1] fig.2).

We modelled a two dimensional grid of hippocampus place cells as a finite state model, where each state x corresponds to a neural activity pattern with one place cell firing and all other place cells silent. For simplicity, we assume that the model rat moves in a grid world and that there is a one-to-one correspondence between the place cells and the grid locations. We also assume that the place cell receptive fields have been learned previously. Each place cell is activated only when the rat visits the corresponding location. In addition we assume that there is a positive continuous variable $\beta(x)$ for each grid location x, as well as one additional global variable $\lambda = \sum_x \beta(x)$.

The exploratory movement of the rat in the grid world induces state transitions in the hippocampus between neighboring subsequent states x_{t-1} and x_t, which implicitly defines the uncontrolled dynamics $q(x_t|x_{t-1})$ (north east south west). When the rat visits location x_{t-1} followed by location x_t, $\beta(x_{t-1})$ changes according to Eq. 6. The result is shown in Fig. 1 for a 15×15 grid with 4 food locations. In the simulation the rat moves 8000 steps randomly according to the uncontrolled dynamics and executing the learning rule Eq. 6 at each iteration. After $800, 4000$ and 8000 steps, the quality of the estimated controlled dynamics Eq. 2 is tested by simulating a single trajectory starting at the central location with coordinates $(8, 8)$. The simulation shows that after about 4000 steps the rat has built an attractor dynamics that converges to one of the food targets.

4 Generalization and neural implementation

In this paper, we have employed a very simple finite state Markov model for the hippocampus place cell activity. Extension to a more realistic model can be easily achieved. One can view the finite state representation of the place cell activity as an abstraction of a cognitive map [2], ie. a topologically organised neural network model (or 'continuous attractor' [7]) with short range excitatory interaction and long range inhibitory interaction. It is well known that in such models with proper parameter settings the neural activity is given by a single localised 'blob' of active neurons and all remaining neurons are silent. The location of the blob is determined in competition by the external sensory input on all neurons. Such an internal map can be used as an abstract preconfigured model of space and place field locations would be associated to sensory landmarks in new environments through learning [8].

There are several possible ways to represent the $\beta(x)$. Since the role of β is similar to the value function in the actor-critic architecture, a possible candidate structure is the ventral striatum [9]. It is an open issue how the optimal control computation Eq. 2 is implemented neurally.

KL learning as presented in this paper only updates at each iteration the value $\beta(x)$ for the state x that is visited at that iteration. It does not require a neural implementation of the uncontrolled dynamics $q(y|x)$ and the state transitions in the hippocampus are induced by the actual behaviour of the animal. The exploratory behaviour according to $q(y|x)$ could be generated in another part of the brain. As a result, the learning is somewhat slow. One can however easily conceive extensions of KL learning that would be significantly faster. When a neural representation of $q(y|x)$ and the immediate reward $V(x)$ exists in the hippocampus, it is possible to 'parallelise' the basic learning algorithm. The parallel computation of the extreme eigenpair becomes essentially the power method: $\beta_{t+1} = H\beta_t, t = 1, \ldots,$ with a normalisation of β_t at each iteration. The power method has exponential convergence independent of problem size with error decreasing as $\left(\frac{|\lambda_2|}{\lambda_1}\right)^t$. A realistic neural implementation may not take full advantage of parallel updating, but even a partially parallel implementation should give a significant acceleration.

5 Discussion

In this paper we have presented KL learning, a novel stochastic method to find the optimal control solution for an average cost KL control problem, and illustrate its applicability to learn goal directed behavior to remembered targets in hippocampus. The result of learning is that the animal can 'think ahead' using the controlled stochastic dynamics $p(y|x)$ that implements an attractor dynamics to one or several of the previously visited goal locations. The results are qualitatively in agreement with the recent findings of [1].

KL learning is a novel stochastic method to find the optimal control solution for an average cost KL control problem. We have discussed how KL learning converges to the extreme eigenvector for the case where the uncontrolled dynamics has a state-independent stationary distribution. Proof of convergence and the treatment of the general case is subject of a future publication.

Alternatively, one could employ one of the mentioned reinforcement learning approach to model these findings. However, KL learning is simpler than reinforcement learning both in terms of required storage and computation: it is simpler than policy improvement-based methods because its computational complexity is essentially equal to a single policy evaluation. It is also simpler than action-value based methods such as Q learning or variants thereof, because it requires computation and storage of a state dependent value function rather than a state-action dependent value function.

One aspect that may lead to a testable difference between RL and KL control is that the optimal KL control policy is probabilistic as can be seen from Eq. 2 whereas for RL the optimal policy is always deterministic [1]. Thus, when an animal is faced with two alternatives with different expected reward, the KL control solution will in general display a 'matching' behaviour where the probability of both action choices is non-zero. The deterministic behaviour is obtained as a special case in the KL control formulation when (a parameter multipying) the reward becomes infinite (or, equivalently, the KL term is zero).

KL learning is a generalization of Z learning, which requires that the extreme eigenvalue is one [5]. This condition is true when the environment has one or more absorbing states located at the goal location(s). This implies that the uncontrolled dynamics (as introduced below) must implement such absorbing states and thus must explicitly depend on the goal location. Instead, KL learning may assume an uncontrolled dynamics that implements a generic exploration of space independently of reward locations.

References

[1] Brad E Pfeiffer and David J Foster. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 2013.

[2] John O'keefe and Lynn Nadel. *The hippocampus as a cognitive map*, volume 3. Clarendon Press Oxford, 1978.

[3] Matthijs AA Van Der Meer and A David Redish. Expectancies in decision making, reinforcement learning, and ventral striatum. *Frontiers in Neuroscience*, 4, 2010.

[4] DJ Foster, RGM Morris, Peter Dayan, et al. A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus*, 10(1):1–16, 2000.

[5] E. Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1369–1376. MIT Press, Cambridge, MA, 2007.

[6] H.J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning Journal*, 2012. DOI 10.1007/s10994-012-5278-7.

[7] Alexei Samsonovich and Bruce L McNaughton. Path integration and cognitive mapping in a continuous attractor neural network model. *The Journal of Neuroscience*, 17(15):5900–5920, 1997.

[8] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[9] Carien S Lansink, Pieter M Goltstein, Jan V Lankelma, Bruce L McNaughton, and Cyriel MA Pennartz. Hippocampus leads ventral striatum in replay of place-reward information. *PLoS biology*, 7(8):e1000173, 2009.

[1] Often, non-deterministic suboptimal RL policies are considered for the purpose of exploration.

CAPI: Generalized Classification-based Approximate Policy Iteration

Amir-massoud Farahmand
School of Computer Science
McGill University
Montreal, Quebec, Canada

Doina Precup
School of Computer Science
McGill University
Montreal, Quebec, Canada

André M.S. Barreto
School of Computer Science
McGill University
Montreal, Quebec, Canada

Mohammad Ghavamzadeh
INRIA Lille - Team SequeL
Lille, France

Abstract

Efficient methods for tackling large reinforcement learning problems usually exploit regularities, or intrinsic structures, of the problem in hand. Most current methods benefit from the regularities of either value function or policy, but not both. In this paper, we introduce a general classification-based approximate policy iteration (CAPI) framework, which can benefit from both types of regularities. This framework has two main components: a generic user-specified value function estimator and a weighted classifier that learns a policy based on the estimated value function. The result is a flexible and sample-efficient class of algorithms.

We also use a particular instantiation of CAPI to design an adaptive treatment strategy for HIV-infected patients. Comparison with a state-of-the-art purely value-based reinforcement learning algorithm, Tree-based Fitted Q-Iteration, shows that benefitting from the regularity of both policy and value function can lead to better performance.

Keywords: Reinforcement Learning, Approximate Policy Iteration, Classification-based Approximate Policy Iteration, HIV Drug Scheduling

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council (NSERC).

1 Introduction

Efficient methods for tackling large reinforcement learning (RL) problems [1] usually exploit special regularities (or structures) of the problem. For example, value-based methods may exploit the smoothness properties of the value function. Many methods have focused on exploiting structure in value function representation and learning, e.g., Farahmand et al. [2], Taylor and Parr [3], Ghavamzadeh et al. [4]. Nonetheless, useful structure can also arise in the policy space. For instance, in many control problems simple policies such as a bang-bang or PID controller can perform quite well if tuned properly. The same is true for many other decision-making problems. In addition to the direct policy search algorithms (e.g., various policy gradient algorithms [5, 6]), one class of methods that try to benefit from the regularities of the policy is the conventional classification-based RL algorithms, e.g., Lagoudakis and Parr [7], Fern et al. [8], Li et al. [9], Lazaric et al. [10]. These methods, however, do not benefit from the regularities of the value function. Thus, they might not be as effective in solving RL problems as one would hope. The goal of this paper is to present a class of algorithms, which we call Classification-based Approximate Policy Iteration (CAPI), that can potentially benefit from the regularities of both value function and policy. This class has been introduced by Farahmand et al. [11] and its theoretical properties have been analyzed there. Here we briefly present the algorithm and describe the application of a particular instantiation of CAPI on the problem of designing an adaptive treatment strategy for HIV-infected patients [12].

Conventional classification-based approaches can be interpreted as a variant of Approximate Policy Iteration (API) that uses rollouts to estimate the action-value function at several points (policy evaluation step) and *projects* the greedy policy obtained at those points onto the predefined space of policies (policy improvement step). In many problems, this approach is helpful because of three main reasons. First, good policies are sometimes simpler to represent and learn than good value functions. Second, even a rough estimate of the value function is often sufficient to separate the best action from the rest, especially when the gap between the value of the greedy action and the rest is large. And finally, even if the best action estimates are noisy (perhaps due to value function imprecision), one can take advantage of powerful classification methods to smooth out the noise. This classification-based approach might particularly be helpful in problems where the classes of "good" policies (parametric or nonparametric) are known in advance, and one simply wants to find good parameters within the class.

Nevertheless, the conventional classification-based RL algorithms suffer from a couple of drawbacks. Most previous work (with the exception of [9, 10, 13, 14] and the related Conservative Policy Iteration approach [15]), uses the $0/1$-loss for training a classifier. This penalizes all mistakes equally and does not consider the relative importance of different regions of the state space. This may lead to surprisingly bad policies [11]. Moreover, the rollout-based estimate of the action-value function does not allow generalization over the state-action space and is quite data-inefficient. This is a big concern in real-world problems where new samples are very expensive or impossible to generate, e.g., adaptive treatment strategy or user dialogue systems. In addition, one cannot easily use rollouts when we only have access to a batch of data, and a generative model or simulator of the environment is not available.

The flexible CAPI framework addresses both issues. The error function used for the classification step in CAPI is weighted according to the difference between the value of the greedy action and those of the other actions. This ensures that the resulting policy closely follows the greedy policy in regions of the state space where the difference between the best action and the rest is considerable (so choosing the wrong action is costly), but pays less attention to regions where all actions are almost the same. Moreover, CAPI allows using any policy evaluation method including, but not restricted to, rollout-based estimates (as in previous work [7, 10]), LSTD [16], policy evaluation version of Fitted Q-Iteration [17, 18], and their regularized variants [2, 19]. This is a significant generalization of existing classification-based RL algorithms, which become special cases of CAPI. This simple change allows us to benefit from the regularities of both the policy and value function, whenever the value and policy spaces are chosen properly. Our theoretical results (reported in [11]) indicate that this extension is indeed sound.

2 Definitions

We consider a *continuous-state, finite-action discounted MDP* $(\mathcal{X}, \mathcal{A}, P, \mathcal{R}, \gamma)$, where $\mathcal{X}$ is a measurable state space (e.g., a subset of $\mathbb{R}^d$), $\mathcal{A}$ is a finite set of actions, P is the transition model, $\mathcal{R}$ is the reward model, and $\gamma \in [0, 1)$ is a discount factor. The mapping $\pi : \mathcal{X} \to \mathcal{A}$ is called a (deterministic) *policy*. As usual, V^π and Q^π denote the value and action-value function for π, while V^* and Q^* denote the corresponding value functions for the optimal policy π^*. A policy π is *greedy* with respect to (w.r.t.) an action-value function Q, denoted by $\pi = \hat{\pi}(\cdot; Q)$, if $\pi(x) = \operatorname{argmax}_{a \in \mathcal{A}} Q(x, a)$ holds for all $x \in \mathcal{X}$.

A recently introduced characterization of the complexity of an RL problem is its *action-gap* regularity [20]. For simplicity, we discuss the action-gap for MDPs with only two actions, i.e., $|\mathcal{A}| = 2$. For any $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, the action-gap function is defined as $\mathbf{g}_Q(x) \triangleq |Q(x, 1) - Q(x, 2)|$ for all $x \in \mathcal{X}$, that is, the difference between the best and second-best action at each state. To understand why the action-gap function is informative, suppose we have an estimate $\hat{Q}^\pi$ of Q^π and we want to perform policy improvement based on $\hat{Q}^\pi$. The greedy policy w.r.t. $\hat{Q}$, i.e., $\hat{\pi}(\cdot; \hat{Q}^\pi)$, should ideally be close

Algorithm 1 CAPI(Π, ν, K)

Input: Policy space Π, State distribution ν, Number of iterations K
Initialize: Let $\pi_{(0)} \in \Pi$ be an arbitrary policy
for $k = 0, 1, \ldots, K-1$ **do**
  Construct a dataset $\mathcal{D}_n^{(k)} = \{X_i\}_{i=1}^n,\ X_i \overset{\text{i.i.d.}}{\sim} \nu$
  $\hat{Q}^{\pi_{(k)}} \leftarrow \text{PolicyEval}(\pi_{(k)})$
  $\pi_{(k+1)} \leftarrow \text{argmin}_{\pi \in \Pi} \hat{L}_n^{\pi_{(k)}}(\pi)$ (action-gap-weighted classification)
end for

to the greedy policy w.r.t. Q^π, i.e., $\hat{\pi}(\cdot; Q^\pi)$. If the action-gap $\mathbf{g}_{Q^\pi}(x)$ is large for some state x, the regret of choosing an action different from $\hat{\pi}(x; Q^\pi)$, roughly speaking, is large; however, confusing the best action with the other one is also less likely. If the action-gap is small, a confusion is more likely to arise, but the regret stemming from the wrong choice will be small. In the next section, we see that the action-gap function is directly used in the formulation of the algorithm. One can prove that the behaviour of the action-gap function for an RL problem influences the difficulty of solving it. This is true for both purely value-based approaches [20] as well as CAPI [11].

3 CAPI Framework

CAPI is an approximate policy iteration framework that takes a policy space Π, a distribution over states $\nu \in \mathcal{M}(\mathcal{X})$, and the number of iterations K as inputs, and returns a policy whose performance should be close to the best policy in Π. Its outline is presented in Algorithm 1. PolicyEval can be any algorithm that computes an estimate $\hat{Q}^\pi$ of Q^π, including: rollout-based estimation (in which case CAPI becomes a nonparametric extension of the DPI algorithm [10]), LSTD-Q [16] and Fitted Q-Iteration [17], or a combination of rollouts and function approximation (in which case CAPI becomes a nonparametric extension of the DPI-Critic algorithm [13] as well as a nonparametric extension of a variant of Classification-based Approximate Modified Policy Iteration of Scherrer et al. [14]).

At each iteration k, the approximate policy improvement step of CAPI is performed by minimizing the following empirical loss function in policy space Π:

$$\hat{L}_n^{\pi_{(k)}}(\pi) \triangleq \int_{\mathcal{X}} \mathbf{g}_{\hat{Q}^{\pi_{(k)}}}(x) \mathbb{I}\{\pi(x) \neq \underset{a \in \mathcal{A}}{\text{argmax}}\, \hat{Q}^{\pi_{(k)}}(x, a)\}\, \mathrm{d}\nu_n, \tag{1}$$

where ν_n is the empirical distribution induced by the samples in $\mathcal{D}_n^{(k)} = \{X_i\}_{i=1}^n$ with $X_i \sim \nu$. The solution to the optimization problem $\pi_{(k+1)} \leftarrow \text{argmin}_{\pi \in \Pi} \hat{L}_n^{\pi_{(k)}}(\pi)$ is the projection of the greedy policy $\hat{\pi}(\cdot; \hat{Q}^{\pi_{(k)}})$, defined only at points $\mathcal{D}_n^{(k)}$, onto policy space Π when the distance measure is weighted according to the estimated action-gap function $\mathbf{g}_{\hat{Q}^{\pi_{(k)}}}$. This should be contrasted with the conventional classification-based approaches [7], which use a uniform weight in all state space, i.e., they minimize $\int_{\mathcal{X}} \mathbb{I}\{\pi(x) \neq \text{argmax}_{a \in \mathcal{A}} \hat{Q}^{\pi_{(k)}}(x, a)\} \mathrm{d}\nu_n$. The statistical properties of CAPI are discussed by Farahmand et al. [11].

As a practical note, the minimization problem $\pi_{(k+1)} \leftarrow \text{argmin}_{\pi \in \Pi} \hat{L}_n^{\pi_{(k)}}(\pi)$ might be difficult to solve for a general policy space Π. The problem is similar (but not identical) to minimizing the $0/1$-loss function in binary classification. A common approach to the non-convex $0/1$-loss optimization is to use a convex surrogate loss, such as action-gap-weighted hinge or exponential loss. This convex relaxation is possible for CAPI too, but we do not formulate it here. Instead, we propose an easy-to-implement solution for this step based on the decision-tree model of classification in the next section.

4 HIV Drug Scheduling

Most of the anti-HIV drugs currently available fall into one of two categories: reverse transcriptase inhibitors (RTI) and protease inhibitors (PI). RTI and PI drugs act differently on the organism, and typical HIV treatments use drug cocktails containing both types of medication. Despite the success of drug cocktails in maintaining low viral loads, there are several complications associated with their long-term use. This has attracted the interest of the scientific community to the problem of optimizing drug-scheduling strategies. Among them, a strategy that has been receiving a lot of attention recently is structured treatment interruption (STI), in which patients undergo alternate cycles with and without the drugs. The scheduling of STI treatments can be seen as a sequential decision-making problem, in which the actions correspond to the types of cocktail that should be administered to a patient [12]. To simplify the problem formulation, it is assumed that RTI and PI drugs are administered at fixed amounts, reducing the available actions to the four possible combinations of drugs. The goal is to minimize the HIV viral load using as little drug as possible.

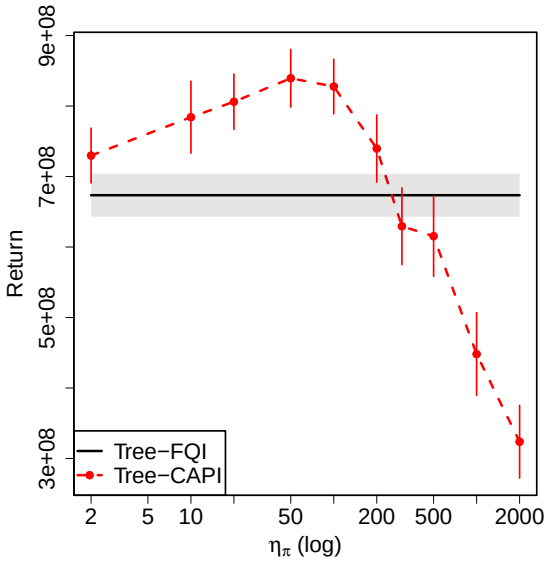


Figure 1: (HIV) Comparing the expected return of Tree-based CAPI vs. Tree-based Fitted Q-Iteration as a function of the complexity of policy space η_π. The policies (in this case STI treatments) were evaluated for $5,000$ days starting from an "unhealthy" state with a high viral load. Error bars and shadowed region show one standard error over 50 runs.

We studied the problem of optimizing STI treatments using a model of the interaction between the immune system and HIV developed by Adams et al. [21] based on real clinical data. All the parameters of the model were set as suggested by Ernst et al. [12]. The methodology adopted in the computational experiments, such as the evaluation of the decision policies and the collection of sample transitions, also followed the protocol proposed by the same authors.

To illustrate the potential benefits of controlling the complexity of the policy space, we compared the pure value-based algorithm adopted by Ernst et al. [12], Fitted Q-Iteration, with CAPI. Following the original experiments, we approximated the value function using an ensemble of 30 decision trees generated by Geurts et al.'s 2006 extra-trees algorithm (we refer to this instantiation of Fitted Q-Iteration as Tree-FQI). Tree-CAPI is similar to Tree-FQI, except that instead of using the greedy policy induced by the current value function approximation, it uses a second ensemble of 30 trees to represent the policy that is the minimizer of (1).

To build the trees representing decision policies, the extra-trees algorithm has to be slightly modified to incorporate the estimated action-gap as its loss function. Such a modification is straightforward, as we now illustrate by showing how a Tree-CAPI policy based on a single tree would perform the action selection. A decision tree defines a partition $\{\mathcal{X}_1, \mathcal{X}_2, \dots\}$ of the state space such that $\bigcup_i \mathcal{X}_i = \mathcal{X}$ and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for $i \neq j$. Define an index function $I : \mathcal{X} \to \{1, 2, \dots\}$ that returns i if $x \in \mathcal{X}_i$. Thus, $\mathcal{D}^{(k)}(x) \triangleq \mathcal{D}_n^{(k)} \cap \mathcal{X}_{I(x)}$ is the set of data points that are in the same partition as x. The Tree-CAPI policy $\pi_{(k+1)}(x)$ is defined as:

$\pi_{(k+1)}(x) \leftarrow \text{argmin}_{a\in\mathcal{A}} \sum_{X_i\in\mathcal{D}^{(k)}(x)} \mathrm{g}_{\hat{Q}^{\pi_k}}(X_i)\mathbb{I}\{a \neq \hat{\pi}(X_i, \hat{Q}^{\pi_{(k)}})\} \equiv \text{argmin}_{a\in\mathcal{A}} \sum_{X_i\mathcal{D}^{(k)}(x)} \hat{Q}^{\pi_{(k)}}(X_i, \hat{\pi}(X_i, \hat{Q}^{\pi_{(k)}})) - \hat{Q}^{\pi_{(k)}}(X_i, a) \equiv \text{argmax}_{a\in\mathcal{A}} \sum_{X_i\in\mathcal{D}^{(k)}(x)} \hat{Q}^{\pi_{(k)}}(X_i, a)$. In the last equality we used the fact that the term $\hat{Q}^{\pi_k}(X_i, \hat{\pi}(X_i, \hat{Q}^{\pi_k}))$ is not a function of a, so it does not influence the minimizer. This is a very simple rule: Pick the action that maximizes the action-value among all the data points in the same partition as x. Note that this is different from choosing the majority over the greedy actions in the partition, which would be the rule if we neglected the action gap. When we have more than a single tree, which is the case when we use extra-trees, the action to be executed is selected by voting, with ties broken randomly.

The complexity of the models built by the extra-trees algorithm can be controlled by the minimum number of points required to split a node during the construction of the trees [22]. In general, the larger this number is, the simpler the resulting models are. Here we fixed this parameter for the trees representing the value function at $\eta_{\mathrm{v}} = 50$, while the corresponding parameter for the policy trees, denoted by η_π, was varied in the set $\{2, 10, 20, 50, 100, 200, 300, 500, 1000, 2000\}$. Figure 1 shows the result of such an experiment.

As shown in Figure 1, restricting the policy space can have a dramatic impact on the performance of the resulting policies. When $2 \leq \eta_\pi \leq 100$ the policies computed by Tree-CAPI perform better than those computed by Tree-FQI, leading to increases on the empirical return as high as 24%. On the other hand, an overly restricted policy space precludes the representation of the intricacies of an efficient STI treatment, resulting in poor performance. With CAPI, it is possible to adjust the complexity of the policy space to a specific context.

5 Conclusion

We proposed CAPI, a general family of classification-based reinforcement learning algorithms, that allows us to design algorithms that benefit from the regularities of both value function and policy. CAPI uses any policy evaluation method, defines an action-gap-weighted loss function, and then finds the policy minimizing this loss, from a desired policy space.

We showed how to efficiently solve the optimization problem (1) for policy spaces induced by a local method such as decision trees. Extending this to other reasonably general classes of policy spaces (e.g., policies that are defined by the sign of linear combination of basis functions) is an open question. Additionally, an interesting research direction is to extend and analyze CAPI for continuous action spaces.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[2] Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized policy iteration. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS - 21)*, pages 441–448. MIT Press, 2009.

[3] Gavin Taylor and Ronald Parr. Kernelized value function approximation for reinforcement learning. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1017–1024, New York, NY, USA, 2009. ACM.

[4] Mohammad Ghavamzadeh, Alessandro Lazaric, Rémi Munos, and Matthew Hoffman. Finite-sample analysis of lasso-TD. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1177–1184, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

[5] Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, pages 319–350, 2001.

[6] Mohammad Ghavamzadeh and Yaakov Engel. Bayesian policy gradient algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 457–464. MIT Press, Cambridge, MA, 2007.

[7] Michail G. Lagoudakis and Ronald Parr. Reinforcement learning as classification: Leveraging modern classifiers. In *ICML '03: Proceedings of the 20th international conference on Machine learning*, pages 424–431, 2003.

[8] Alan Fern, Sungwook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research*, 25:85–118, 2006.

[9] Lihong Li, Vadim Bulitko, and Russell Greiner. Focus of attention in reinforcement learning. *Journal of Universal Computer Science*, 13(9):1246–1269, 2007.

[10] Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Analysis of a classification-based policy iteration algorithm. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 607–614. Omnipress, 2010.

[11] Amir-massoud Farahmand, Doina Precup, and Mohammad Ghavamzadeh. Generalized classification-based approximate policy iteration. In *Tenth European Workshop on Reinforcement Learning (EWRL)*, 2012.

[12] Damien Ernst, Guy-Bart Stan, Jorge Gongalves, and Louis Wehenkel. Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. In *IEEE Conference on Decision and Control*, pages 667–672. IEEE, 2006.

[13] Victor Gabillon, Alessandro Lazaric, Mohammad Ghavamzadeh, and Bruno Scherrer. Classification-based policy iteration with a critic. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, Washington, USA,, 2011. Omnipress.

[14] Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, and Matthieu Geist. Approximate modified policy iteration. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

[15] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pages 267–274, 2002.

[16] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[17] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[18] Amir-massoud Farahmand and Doina Precup. Value pursuit iteration. In *Advances in Neural Information Processing Systems (NIPS - 25)*, 2012.

[19] Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian Decision Problems. In *Proceedings of American Control Conference (ACC)*, pages 725–730, June 2009.

[20] Amir-massoud Farahmand. Action-gap phenomenon in reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS - 24)*, 2011.

[21] B.M. Adams, H.T. Banks, Hee-Dae Kwon, and H.T. Tran. Dynamic multidrug therapies for HIV: optimal and STI control approaches. *Mathematical Biosciences and Engineering*, 1(2):223–41, 2004.

[22] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 36(1):3–42, 2006.

Collecting reward to defend homeostasis: A homeostatic reinforcement learning theory

Mehdi Keramati
Group for Neural Theory
LNC (INSERM U960),
Ecole Normale Superieure, Paris, France
m.mahdi.keramati@gmail.com

Boris Gutkin
Group for Neural Theory
LNC (INSERM U960),
Ecole Normale Superieure, Paris, France
boris.gutkin@ens.fr

Abstract

Survival requires efficient regulation of the internal homeostasis and defending it against perturbations. This in turn calls for complex behavioral strategies for obtaining physiologically depleted resources. In other words, in complex environments, the animal must learn what to do in order to fulfill its needs. To do so it is essential that brain systems monitoring homeostatic integrity as well as systems controlling motivation and implementing behavioral learning through associative mechanisms work in concert. We propose a normative computational theory for homeostatically regulated reinforcement learning, where physiological stability and reward acquisition prove to be identical objectives achievable simultaneously. Theoretically, the framework resolves the long standing question of how an animal motivation is modulated by its internal state and how an animal would learn to predictively act to pre-empt homeostatic challenges. It further provides a normative explanation for temporal discounting of reward, by showing that discounting future rewards is necessary in order to achieve the fundamental objective of defending homeostasis via the reward-seeking mechanism. Moreover, the theory accounts for risk aversive behavior, taste-induced overeating, animals' lack of motivation for intravenous injection of food, and animals' motivation toward foods with no energy content. Neurobiologically, our theory clarifies the formal computational relationship between the hypothalamic orexinergic circuitry, and the midbrain dopaminergic nuclei, signaling as an interface between the internal states and motivated behaviors.

Keywords: Reinforcement Learning; Homeostatic Regulation; Reward; Anticipatory Responding; Temporal Discounting

Acknowledgements

M.K. and B.G. are supported by grants from Frontiers du Vivant, the French MESR, CNRS, INSERM, ANR, ENP and NERF.

1 Background

Survival requires the living organisms to maintain their integrity within the environment. Otherwise said, they must preserve homeostasis. In this sense, efficient reward-seeking decisions depend on two brain circuits working in concert: (1) a "homeostatic regulation (HR)" system that tracks the internal state of the animal, and is concerned with defending it against perturbations around the homeostatic setpoint, and (2) a "reinforcement learning (RL)" process capable of exploiting the experienced environmental contingencies and reward history, in order to make efficient instrumental reward-seeking responses. Communication between these two systems is essential for behavior to be responsive to the mandate of maintaining a stable *internal milieu* and thus survival. However, the computational mechanisms underlying this obvious coupling remains poorly understood, as the two systems have been traditionally studied separately.

The integration of the two systems is behaviorally manifest in the classical behavioral pattern of anticipatory responding: upon observing conditioned stimuli that predict homeostatic perturbations (e.g. a cue that predicts cold air current), animals take preventive measures (e.g. shivering) to preclude disturbances in regulated variables (e.g. body temperature). Here, for the first time to our knowledge, we propose a normative explanation for anticipatory responding.

Neurobiologically, accumulating evidence over the past decade shows intricate intercommunication between the hypothalamus and the brain reward-learning system[1]. The hypothalamic nuclei, constituting the HR system, monitor the internal state of the organism and produce appropriate corrective responses, thereby playing a central role in the homeostatic regulation of feeding and energy metabolism, body temperature, blood osmolality, etc.[1]. The cortico-basal ganglia circuits and the dopaminergic system, on the other hand, are known for their role in Pavlovian, habitual and goal-directed decision making[2]. The neural integration between the RL and HR systems is particularly well-studied for the case of energy regulation, where orexin neurons project from the lateral hypothalamus to the ventral tegmental area (VTA), and modulate the activity of dopamine neurons as a function of the animal's energy state[1]. A key, yet unaddressed question here is what computations, at an algorithmic level, are being performed in this biological integration of the two systems, and how such computations give rise to behavioral patterns like anticipatory responding.

2 Theory

The interaction between drive and incentive has been the subject of much debate in the motivation literature in psychology. Neo-behaviorists like Hull[3], Spence[4] and Mowrer[5] proposed the "drive-reduction" theory of motivation to define the nature of reward. According to this theory, one primary mechanism underlying reward is the usefulness of the corresponding stimulus in fulfilling the homeostatic needs of the organism, as determined by interoceptive signals[6]. Building upon this theory, we derive a formal definition of reward as the ability of a stimulus in counteracting disruptions in the physiological state, and restoring the internal equilibrium.

We start by defining "homeostatic space" as a multidimensional metric space where each dimension represents one regulated physiological variable (the horizontal plane in **Fig 1**). The physiological state of the animal at each time t can be represented as its position in this space, denoted by $H_t = (h_{1,t}, h_{2,t}, .., h_{N,t})$, where $h_{i,t}$ indicates the state of the i-th physiological variable. For example, $h_{i,t}$ can refer to the animal's glucose level, body temperature, plasma osmolality, etc. The homeostatic setpoint, as the ideal internal state, can also be denoted by $H^* = (h_1^*, h_2^*, .., h_N^*)$. As a mapping from physiological state to motivational state, we define animal's "drive" as the distance of the internal state from the setpoint (the three-dimensional surface in **Fig 1**):

$$D(H_t) = \sqrt[m]{\sum_{i=1}^{N} \left|h_i^* - h_{i,t}\right|^n} \quad (1$$

Having defined drive, we can now provide a formal definition for reward, based on the drive reduction theory. Assume that as the result of an action, the animal receives an outcome o_t at time t. The impact of this outcome on different dimensions of the animal's internal state can be denoted by $K_t = (k_{1,t}, k_{2,t}, .., k_{N,t})$. For example, $k_{i,t}$ can be the quantity of glucose received, as the outcome o_t was consumed. Consumption of such outcome will result in a transition of the physiological state from H_t to $H_{t+1} = H_t + K_t$ (see **Fig 1**) and consequently, a transition of the drive state from $D(H_t)$ to $D(H_{t+1}) = D(H_t + K_t)$. Accordingly, the rewarding value of this outcome can be defined as the consequent reduction in the drive function:

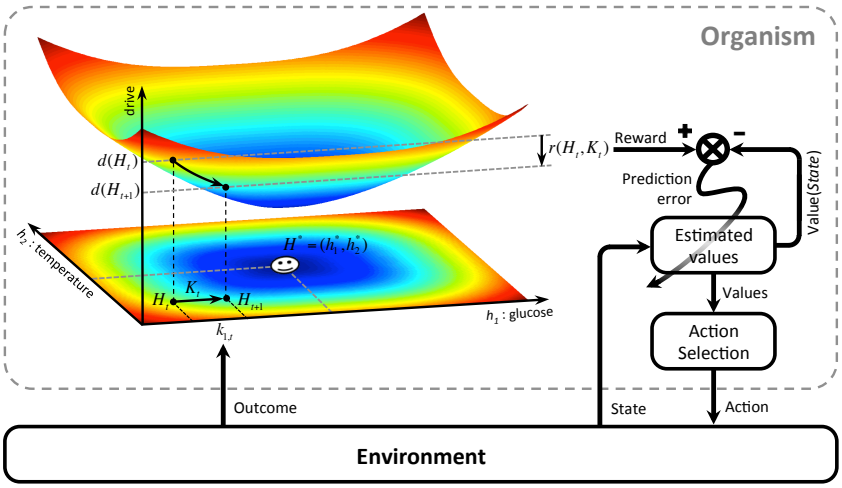


Fig 1: Schematics of the model.

$$r(H_t, K_t) = D(H_t) - D(H_{t+1}) = D(H_t) - D(H_t + K_t) \quad (2$$

Intuitively, the rewarding value of an outcome depends on the ability of its constituting elements in reducing the homeostatic distance from the setpoint. This internal state-dependent reward can then be used by any variant of the RL algorithm, namely model-free RL (as in **Fig 1**), model-based RL[7], hierarchical RL[8], etc.

3 Analytical Results

Theoretically, this definition of reward reconciles the RL and HR theories in terms of their normative assumptions: It can be shown that reward acquisition and physiological stability are mathematically equivalent objective functions (see [9] for the proof). More precisely, given the proposed definition of reward, any behavioral policy, π, that maximizes the sum of discounted rewards (SDR) is at the same time minimizing the sum of discounted deviations (SDD) from the setpoint, and vice versa:

$$if\ \gamma < 1: \quad \underset{\pi}{\operatorname{argmin}}\, SDD(\pi) = \underset{\pi}{\operatorname{argmax}}\, SDR(\pi) \quad (3$$

In this respect, reward acquisition learned by the RL system can be seen as means that guide animal's behavior toward satisfying the more basic objective of defending homeostasis.

The equivalency of reward maximization and deviation minimization objectives in our model depends on a critical requirement: $\gamma < 1$, i.e. future expected rewards and future predicted homeostatic deviations should be discounted (see [9]). In fact, if there is no discounting ($\gamma = 1$), then the rewarding value of behavioral policies that change the position of the internal state only depends on the initial and final internal states, regardless of its trajectory in the homeostatic space. In the absence of discounting, the values of two behavioral policies with equal net shifts of the internal state are equal, even if one policy moves the internal state along the shortest path, whereas the other policy results in large deviations of the internal state from the setpoint and thus, threatens survival. These results hold not only for exponential discounting, but also under hyperbolic discounting. In this respect, our theory gives a normative explanation for the necessity of temporal discounting of reward: to fulfill the basic objective of internal stability, it is necessary to discount future rewards.

4 Behavioral Results

A wide range of evidence shows that animals make anticipatory responses to preclude perturbations in regulated variables, even before any physiological depletion (negative feedback) is detectable. As one clear example, progressive tolerance to alcohol-induced hypothermia in rats is shown to be mediated by associative learning processes (**Fig. 2**). In fact, the animal gradually learns to initiate a tolerance response

upon observing a stimulus that predicts infusion of ethanol, in order to minimize the ethanol-induced decrease of body temperature (**Fig. 2**).

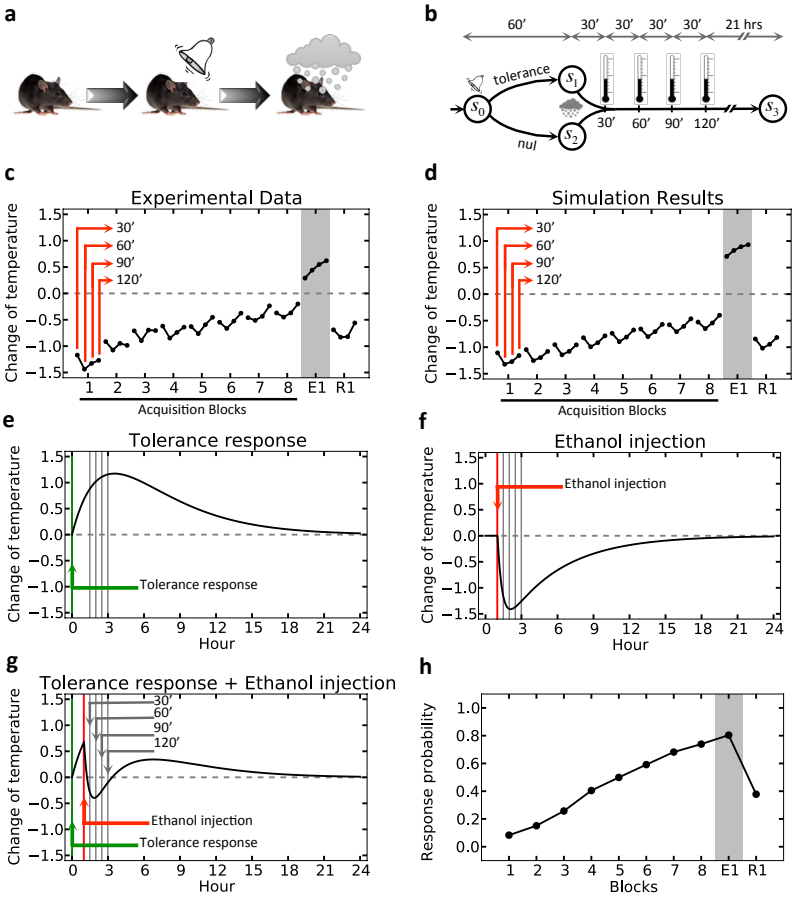


Figure 2. Experimental (Adapted from[10]) and simulation results of acquisition and extinction of conditioned tolerance response to ethanol. **(a)** In each block (day) of the experiment, the animal receives ethanol injection after the presentation of the stimulus. **(c)** In every block, the change in the body temperature is measured 30, 60, 90, and 120 minutes after ethanol administration. (**b**) The model is simulated in an artificial task where the agent has the choice between doing nothing and triggering the tolerance response, upon observing the stimulus. (**d**) Simulation results replicate experimental data. Plots (**e**) and (**f**) show the assumed dynamics of body temperature upon initiation of tolerance response and ethanol administration, respectively. Plot (**g**) shows the combined effect, which results in less deviation from the setpoint, compared to the two other cases. Plot (**h**), which is averaged over 500 simulated agents, shows that the model learns gradually that the optimal strategy is to trigger the tolerance response upon observing the stimulus. E1: the first block of extinction; R1: the first block of re-acquisition.

Although the utility of anticipatory responding is intuitively obvious, to the best of our knowledge, no "formal" normative explanation had been proposed for it so far. The explanation we proposed above is normative, as selecting the tolerance response over the alternative option (null) maximizes reward, and also minimizes homeostatic deviation (compare **Fig. 2g** and **Fig. 2f**).

5 Neural Substrates

Orexin neurons, originating predominantly in the lateral hypothalamus area, project to several brain regions including the VTA[11]. Orexin neurons are responsive to peripheral metabolic signals, as well as the animal's deprivation level[12], as they are innervated by orexigenic and anorexigenic neural populations in the arcuate nucleus of hypothalamus where circulating peptides are sensed. Data shows that orexin agonists induce feeding behavior, and orexin antagonist reduces appetite for food[13,14], thereby defining the active role for these neurons in food-seeking behavior. Interestingly, orexin projections to the VTA are shown to have an excitatory effect on the firing activity of dopaminergic neurons[15,16], placing these at the interface between internal states and the reward learning circuit[1,17]. As in our model the internal deprivation signal modulates the rewarding value of food and thus the prediction error signal, it provides a normative explanation for the role of orexin neurons in modulating the phasic activity of dopamine neurons.

6 Discussion

Critically, our model is built upon the drive-reduction theory of motivation, initially proposed by Clark Hull[3], and became the dominant theory of motivation in psychology during 1940s and 1950s. This theory, however, is largely abandoned today, due to a number of criticisms that had questioned its validity. The most important problem is that it does not explain how secondary reinforcers (e.g. money, or a light that predicts food) would gain motivational value, since they don't reduce any drive, *per se*. Our model resolves this problem as the RL algorithms considered in our framework propagate the drive reduction-induced reward of primary reinforcers to secondary reinforcers that predict them, as well as to behavioral policies that lead to them. The second major criticism is that the classical drive-reduction theory has trouble

accounting for why in some cases animals voluntarily increase their drive, even in the absence of any physiological deficit (e.g. shivering under normal body temperature). Our model alleviates this criticism by accounting for anticipatory responding. Thirdly, the drive-reduction theory seems in contradiction with the evidence showing that intravenous injection (and even intragastric intubation, in some cases) of food is not rewarding, despite the fact that its drive-reduction effect is equal to when that food is ingested orally[18]. Finally, the classical drive-reduction theory does not explain why animals show strong motivation toward some foods that have no nutritional content (e.g. saccharine) and thus, no drive-reduction effect. These two criticisms can be resolved in our framework by simply assuming that orosensory properties of food provide the animals with an estimate, $\hat{k}_{i,t}$, of its true nutritional content, $k_{i,t}$. Based o this assumption, experiencing the orosensory properties of food is necessary for computing its reinforcing effect. Also, erroneous approximation of the drive-reduction ability of hyperpalatable food produces motivation toward them that is decoupled from their caloric content.

7 References

1. Palmiter, R. D. Is dopamine a physiologically relevant mediator of feeding behavior? *Trends in neurosciences* **30**, 375–81 (2007).
2. Balleine, B. W. & O'Doherty, J. P. Human and rodent homologies in action control: corticostriatal determinants of goal-directed and habitual action. *Neuropsychopharmacology* **35**, 48–69 (2010).
3. Hull, C. L. *Principles of behavior: an introduction to behavior theory*. (Appleton-Century-Crofts: New York, 1943).
4. Spence, K. W. *Behavior theory and conditioning*. (Greenwood Press: Westport, 1956).
5. Mowrer, O. H. *Learning theory and behavior*. (Wiley: New York, 1960).
6. Cabanac, M. Physiological Role of Pleasure. *Science* **173**, 1103–1107 (1971).
7. Daw, N. D., Niv, Y. & Dayan, P. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience* **8**, 1704–1711 (2005).
8. Botvinick, M. M. Hierarchical models of behavior and prefrontal function. *Trends in cognitive sciences* **12**, 201–8 (2008).
9. Keramati, M. & Gutkin, B. A reinforcement learning theory for homeostatic regulation. *Advances in Neural Information Processing Systems 24* 82–90 (2011).
10. Mansfield, J. G. & Cunningham, C. L. Conditioning and extinction of tolerance to the hypothermic effect of ethanol in rats. *Journal of Comparative and Physiological Psychology* **94**, 962–969 (1980).
11. Sakurai, T. *et al.* Orexins and orexin receptors: a family of hypothalamic neuropeptides and G protein-coupled receptors that regulate feeding behavior. *Cell* **92**, 573–585 (1998).
12. Burdakov, D., Gerasimenko, O. & Verkhratsky, A. Physiological changes in glucose differentially modulate the excitability of hypothalamic melanin-concentrating hormone and orexin neurons in situ. *The Journal of Neuroscience* **25**, 2429–2433 (2005).
13. Yamanaka, A., Sakurai, T., Katsumoto, T., Yanagisawa, M. & Goto, K. Chronic intracerebroventricular administration of orexin-A to rats increases food intake in daytime, but has no effect on body weight. *Brain research* **849**, 248–52 (1999).
14. Rodgers, R. J. *et al.* SB-334867, a selective orexin-1 receptor antagonist, enhances behavioural satiety and blocks the hyperphagic effect of orexin-A in rats. *The European journal of neuroscience* **13**, 1444–52 (2001).
15. Korotkova, T. M., Sergeeva, O. A., Eriksson, K. S., Haas, H. L. & Brown, R. E. Excitation of ventral tegmental area dopaminergic and nondopaminergic neurons by orexins/hypocretins. *The Journal of Neuroscience* **23**, 7–11 (2003).
16. Narita, M. *et al.* Direct involvement of orexinergic systems in the activation of the mesolimbic dopamine pathway and related behaviors induced by morphine. *The Journal of neuroscience* **26**, 398–405 (2006).
17. Scammell, T. E. & Saper, C. B. Orexin, drugs and motivated behaviors. *Nature Neuroscience* **8**, 1286–1288 (2005).
18. Miller, N. E. & Kessen, M. L. Reward effects of food via stomach fistula compared with those of food via mouth. *Journal of Comparative and Physiological Psychology* **45**, 555–564 (1952).

Trial-based Heuristic Tree Search for Finite Horizon MDPs

Thomas Keller
University of Freiburg
Freiburg, Germany
tkeller@informatik.uni-freiburg.de

Malte Helmert
University of Basel
Basel, Switzerland
malte.helmert@unibas.ch

Abstract

Dynamic programming is a well-known approach for solving MDPs. In large state spaces, asynchronous versions like Real-Time Dynamic Programming (RTDP) have been applied successfully. If unfolded into equivalent trees, Monte-Carlo Tree Search algorithms are a valid alternative. UCT, the most popular representative, obtains good anytime behavior by guiding the search towards promising areas of the search tree and supporting non-admissible heuristics. The global Heuristic Search algorithm AO* finds optimal solutions for MDPs that can be represented as acyclic AND/OR graphs.

Despite the differences, these approaches actually have much in common. We present the Trial-based Heuristic Tree Search (THTS) framework that subsumes these approaches and distinguishes them based on only five ingredients: heuristic function, backup function, action selection, outcome selection, and trial length. We describe the ingredients that model RTDP, AO* and UCT within this framework, and use THTS to combine attributes of these algorithms step by step in order to derive novel algorithms with superior theoretical properties. We merge Full Bellman and Monte-Carlo backup functions to Partial Bellman backups, and gain a function that both allows partial updates and a procedure that labels states when they are solved. DP-UCT combines attributes and theoretical properties from RTDP and UCT even though it differs from the latter only in the used Partial Bellman backups. Our main algorithm, UCT* adds a limited trial length to DP-UCT to inherit the global search behavior of AO* , which ensures that parts of the state space that are closer to the root are investigated more thoroughly. The experimental evaluation shows that both DP-UCT and UCT* are not only superior to UCT, but also outperform PROST, the winner of the International Probabilistic Planning Competition (IPPC) 2011 on the benchmarks of IPPC 2011.

Keywords: Planning under Uncertainty, MDPs

Acknowledgements

Thomas Keller is supported by the German Aerospace Center (DLR) as part of the Kontiplan project (50 RA 1221).

1 Introduction

Markov decision processes (MDPs) offer a general and widely used framework for decision making under uncertainty. Among the algorithms that have been applied successfully to MDPs with large state spaces are the asynchronous Dynamic Programming algorithm RTDP (Barto et al., 1995), the Heuristic Search approach AO* and UCT (Kocsis and Szepesvári, 2006), the most a popular representative of Monte-Carlo Tree Search (MCTS) (Browne et al., 2012).

The wide variety of possible choices when it comes to solving MDPs gives rise to the question which differences and commonalities exist among them. We answer this question by introducing Trial-based Heuristic Tree Search (THTS), a framework that subsumes all of these algorithms (Keller and Helmert, 2013). Specific instances of THTS can be described with only five ingredients: heuristic function, backup function, action selection, outcome selection, and trial length. We furthermore use THTS to combine ingredients of UCT, RTDP and AO* to derive algorithms that are stronger than the individual methods and even outperform PROST (Keller and Eyerich, 2012), the winner of the International Probabilistic Planning Competition (IPPC) 2011 on the benchmarks of IPPC 2011.

2 Background

In this paper, we are interested in algorithms for finite horizon MDPs S$\langle S, A, P, R, H, s_0\rangle$ with the usual semantics (Puterman, 1994) that have access to a declarative model of transition probabilities and rewards. Usually, a solution for such an MDP is a policy, i.e. a mapping from states to actions. As a policy is already expensive to describe (let alone compute) in MDPs with large state spaces, we consider algorithms that do not generate the policy offline but interleave planning of a single action and its execution, a process that is repeated H times. In the first step of a run, s_0 is set to a given initial state. In each other step, it is set to the outcome of the last action execution. We estimate the quality of an algorithm by sampling a fixed number of such runs. This approximates the expected reward of a policy π in MDP M, as the exact calculation is also intractable in large MDPs. As it is important for our notion of anytime optimal backup functions, we define the expected reward in terms of the state-value function V^π as $V^\pi(M) := V^\pi(s_0)$ with

$$V^\pi(s) := \begin{cases} 0 & \text{if } s \text{ is terminal} \\ Q^\pi(\pi(s), s) & \text{otherwise,} \end{cases}$$

where the action-value function $Q^\pi(a, s)$ is defined as

$$Q^\pi(a, s) := R(a, s) + \sum_{s' \in S} P(s'|a, s) \cdot V^\pi(s').$$

The optimal policy π^* in M can be derived from the related Bellman optimality equation (Bellman, 1957), which describes the reward for selecting the actions that yield the highest expected reward.

We define states such that the number of remaining steps is part of a state, and denote it with $s[h]$ for a state $s \in S$. A state with $s[h] = 0$ is called terminal. As the number of remaining steps must decrease by one in each state transition, i.e. $P(s'|a, s) = 0$ if $s'[h] \neq s[h]-1$, each finite-horizon MDP induces a directed acyclic graph. Moreover, we require that all policies are proper in M, a constraint that is satisfied in all benchmark domains of IPPC 2011.

3 Trial-based Heuristic Tree Search

In this section, we give a brief overview on the Trial-based Heuristic Tree Search framework. For a detailed description, we refer the reader to the full version of this paper (Keller and Helmert, 2013). The THTS framework bridges the gap between Dynamic Programming, MCTS, and Heuristic Search algorithms and can be used to model most members of these families of algorithms. THTS algorithms maintain an explicit tree of alternating decision and chance nodes. Decision nodes are tuples $n_d = \langle s, V^k\rangle$, where $s \in S$ is a state and $V^k \in \mathbb{R}$ is the state-value estimate based on the k first trials. Chance nodes are tuples $n_c = \langle s, a, Q^k\rangle$, where $s \in S$ is a state, $a \in A$ is an action, and $Q^k \in \mathbb{R}$ is the action-value estimate based on the k first trials. In the following, we denote decision nodes with n_d, chance nodes with n_c, and the set of successor nodes of n in the explicit tree with $\mathcal{S}(n)$. We abbreviate $R(s(n_c), a(n_c))$ with $R(n_c)$ and $P(s(n_d)|a(n_c), s(n_c))$ with $P(n_d|n_c)$.

Initially, the explicit tree contains only the root node n_0, a decision node with $s(n_0) = s_0$. Each trial, which can be separated in three phases, adds nodes to the tree (it explicates them). In the selection phase, the explicit tree is traversed by alternatingly choosing one or more successor nodes according to action and outcome selection. When a previously unvisited decision node is encountered, the expansion phase starts. A child is added to the explicit tree for each applicable action (or for each outcome of each applicable action) and all estimates are initialized with a heuristic. That way, all successor nodes of the currently visited node contain action-value estimates, so THTS algorithms may switch back to the selection phase, a process that continues until the desired trial length is reached (which is given when an empty

set of successors is selected). All visited nodes are updated in reverse order in the subsequent backup phase, possible interrupted by additional selection and expansion phases if multiple actions or outcomes have been selected earlier. The trial finishes when the backup function is called on the root node. Another trial is started unless the root node is solved or a time constraint is met..

Algorithm 1: The THTS schema.

```
1  THTS(MDP M, timeout T):
2      n_0 ← getRootNode(M)
3      while not solved(n_0) and time() < T do
4          visitDecisionNode(n_0)
5      return greedyAction(n_0)
6  visitDecisionNode(Node n_d):
7      if n_d was never visited then initializeNode(n_d)
8      N ← selectAction(n_d)
9      for n_c ∈ N do
10         visitChanceNode(n_c)
11     backupDecisionNode(n_d)
12 visitChanceNode(Node n_c):
13     N ← selectOutcome(n_c)
14     for n_d ∈ N do
15         visitDecisionNode(n_d)
16     backupChanceNode(n_c)
```

An THTS algorithm is specified in terms of five ingredients: heuristic function, backup function, action selection, outcome selection, and trial length. We focus on different backup functions in the following due to space constraints, and refer to the full version of this work for a detailed overview on possible choices for the other ingredients. To get an intuition, we introduce popular choices in a nutshell. Action selection is a popular research area which can mostly be divided in two camps. On the one hand are algorithms like AO* or RTDP that always select the successor node with the highest action-value estimate. Combined with an admissible heuristic this can enable considerable pruning effects. On the other hand are action selection strategies that balance exploration and exploitation. One such example is UCT, where the UCB1 formula (Auer et al., 2002) is used to select actions that have been rarely tried or yielded promising results in the past. This allows the usage of non-admissible heuristic functions which are usually cheaper to compute yet similarly informative. Outcome selection is the ingredient with the least variance in well-studied algorithms – almost all algorithms select a chance node according to its probability (possibly biased by solved siblings).

The backup function defines how the information that is gathered in trials is propagated through the tree. In the THTS framework, nodes are updated only based on values of some or all of their successor nodes. In this work, we focus on algorithms that are anytime optimal, i.e. yield reasonable results quickly, improve when more trials are available and eventually converge towards the optimal decision (w.r.t. the expected reward). One way to guarantee optimality in the limit is to demand that the backup function is such that estimates converge towards the optimal value functions, i.e. $Q^k(n_c) \to Q^*(a(n_c), s(n_c))$ and $V^k(n_d) \to V^*(s(n_d))$ for all n_d, n_c and $k \to \infty$. We distinguish between full and partial backup functions. Full backup functions can only be computed for a node if each of its children in the MDP is also represented in the explicit tree, i.e. $\mathcal{S}(n_d) = \{\langle s(n_d), a, Q^k\rangle | a \in A\}$ and $\sum_{n_d \in \mathcal{S}(n_c)} P(n_d|n_c) = 1$. Partial backup functions require only one explicated child, so action-value estimates can be calculated even if only one outcome is in the explicit tree (and the one that was selected in the current trial always is). We focus on the development of such a backup function as we consider MDPs with a potentially huge number of outcomes (up to 2^{50} in our experiments).

Monte-Carlo backup MCTS algorithms like UCT are based on Monte-Carlo backups, a partial backup function which extends the current average with the latest sampled value (Sutton and Barto, 1998). As probabilities of outcomes are not used to calculate Monte-Carlo backups, they are the predominant backup function in learning scenarios where only a generative model of the MDP is available. Let $C^k(n)$ be the number of times node n has been visited among the first k trials. State-value estimate and action-value estimate are calculated with Monte-Carlo backups as

$$V^k(n_d) = \begin{cases} 0 & \text{if } s(n_d) \text{ is terminal} \\ \frac{\sum_{n_c \in \mathcal{S}(n_d)} C^k(n_c) \cdot Q^k(n_c)}{C^k(n_d)} & \text{otherwise,} \end{cases}$$

$$Q^k(n_c) = R(n_c) + \frac{\sum_{n_d \in \mathcal{S}(n_c)} C^k(n_d) \cdot V^k(n_d)}{C^k(n_c)}.$$

For Monte-Carlo backups to converge towards the optimal value function, the outcome selection strategy must be s.t. $\frac{C^k(n_d)}{C^k(n_c)} \to P(n_d|, n_c)$ for $k \to \infty$, and the action selection strategy s.t. $\frac{C^k(n_c^*)}{C^k(n_d)} \to 1$ for $k \to \infty$, where $n_c^* = \langle s, \pi^*(s), Q^k\rangle$ is the successor of n_d in the optimal policy π^*. It is often not trivial to prove that Monte-Carlo backups converge, but it was shown by Kocsis and Szepesvári (2006) for the UCT algorithm.

Full Bellman backup Another prominent method to propagate information in the tree are Full Bellman backups:

$$V^k(n_d) = \begin{cases} 0 & \text{if } s(n_d) \text{ is terminal} \\ \max_{n_c \in \mathcal{S}(n_d)} Q^k(n_c) & \text{otherwise,} \end{cases}$$

$$Q^k(n_c) = R(n_c) + \sum_{n_d \in \mathcal{S}(n_c)} P(n_d|n_c) \cdot V^k(n_d).$$

As the name implies, they are the full backup function derived from the Bellman optimality function. Unlike Monte-Carlo backups, Full Bellman backups allow a procedure that labels nodes as solved. Obviously, each algorithm that is based on Full Bellman backups and selects actions and outcomes among unsolved nodes is anytime optimal, as all states will eventually be visited with $k \to |S|$.

Max Monte-Carlo backup Both presented backup functions have their advantages and disadvantages. On our way to combine the desirable parts of both, Max-Monte-Carlo backups are a first step. They use the action-value backup function of Monte-Carlo and the state-value backup function of Full Bellman backups. In other words, a decision node is updated based on the value of its best child rather than aggregating over all children.

Like Monte-Carlo backups, Max-Monte-Carlo backups are both partial and do not rely on a generative model of the MDP. But they are also much more resilient for scenarios where different actions yield vastly different expected rewards than their counterpart where decision nodes are updated by aggregating over all successors. Our first algorithm, **MaxUCT**, uses the same ingredients as UCT, but updates nodes with the Max-Monte-Carlo backup function.

Partial Bellman backup In a second step, we aim to derive a backup function that additionally supports a solve labeling procedure. Therefore, it is necessary to exploit the declarative model by considering probabilities in the backups of action-value estimates. Or, from a different point of view, we are looking for a partial version of Full Bellman backups that does not require all successor nodes to be explicated. To calculate an estimate, we weight the outcomes that are part of the tree proportionally to their probability and to the missing outcomes:

$$V^k(n_d) = \begin{cases} 0 & \text{if } s(n_d) \text{ is terminal} \\ \max_{n_c \in \mathcal{S}(n_d)} Q^k(n_c) & \text{otherwise,} \end{cases}$$

$$Q^k(n_c) = R(n_c) + \frac{\sum_{n_d \in \mathcal{S}(n_c)} P(n_d|n_c) \cdot V^k(n_d)}{P^k(n_c)},$$

where $P^k(n_c) = \sum_{n_d \in \mathcal{S}(n_c)} P(n_d|n_c)$ is the sum of the probabilities of all outcomes that are explicated. Intuitively, we expect estimates of outcomes that are not part of the explicit tree to be comparable to the probability weighted outcomes that are explicated. Partial Bellman backups converge towards the Bellman optimality equation under selection strategies that explore the whole tree, as $P^k(n_c) \to 1$ and $Q^k(n_c) \to Q^*(a(n_c), s(n_c))$ for $k \to |S|$.

Our second algorithm, **DP-UCT**, is also identical to UCT except for the backup function. It combines properties of Dynamic Programming and UCT. It resembles MCTS as it incorporates the advantages of partial backup functions, balanced action selection and the usage of non-admissible heuristic functions. The fact that the backup function takes probabilities into account and allows solve labeling and termination when the root node is solved is just like in Dynamic Programming approaches. To our knowledge, this theoretical property was never incorporated into an algorithm that selects actions based on the UCB1 formula.

DP-UCT benefits a lot from the balanced action selection strategy at the beginning of each trial. As the uncertainty grows with the number of simulated steps, states that are far from the root often have only little influence on that decision, even if they are part of the optimal solution or crucial for some future decision. Therefore, investigating the state space close to the root more thoroughly might improve action selection with short time windows. This idea is incorporated in our main algorithm, **UCT***, which is identical to DP-UCT except that is uses a property inherent to Heuristic Search. These global search algorithms finish a trial whenever a tip node is expanded – a natural way to focus the search on states close to the root that maintains optimality in the limit. UCT* still produces the asymmetric search trees that spend more time in promising parts of the tree, but it also makes sure that it takes the time to investigate parts that turn out to be different than what they looked like at first glance.

4 Experimental Evaluation

We evaluate the algorithms MaxUCT, DP-UCT and UCT by performing experiments on 2.66 GHz Intel Quad-Core Xeon computers, with one task per core simultaneously and a memory limit of 2.5 GB. We use the IPPC 2011 benchmarks, a set of eight domains with ten problems each. Each instance is a finite-horizon MDP with a horizon of 40. Each algorithm is evaluated based on 100 runs on each problem rather than the 30 of IPPC 2011 to obtain statistically significant results. The results in Table 1 are achieved with a timeout of 2 seconds per decision. They are obtained by converting the averaged accumulated rewards of the 100 runs to relative scores on a scale from 0 to 1 for each problem, and then calculating the average over these values from all instances for each domain. A relative score of 0 is assigned to an artificial minimum policy taken from IPPC 2011, and the score of 1 is given to the planner that achieved the highest reward. The total result is calculated as the average over all domains. Additional experiments are analyzed in the full version of this work (Keller and Helmert, 2013),

	ELEVATORS	SYSADMIN	RECON	GAME	TRAFFIC	CROSSING	SKILL	NAVIGATION	Total
UCT	0.93	0.66	**0.99**	0.88	0.84	0.85	0.93	0.81	0.86
MaxUCT	**0.97**	0.71	0.88	0.9	0.86	**0.96**	0.95	0.66	0.86
DP-UCT	**0.97**	0.65	0.89	0.89	0.87	**0.96**	**0.98**	**0.98**	0.9
UCT*	**0.97**	**1.0**	0.88	**0.98**	**0.99**	**0.98**	**0.97**	**0.96**	**0.97**
PROST	0.93	0.82	**0.99**	0.93	0.93	0.82	**0.97**	0.55	0.87

Table 1: Score per domain and total scores for the IPPC 2011 benchmarks. Best results (± 0.02) are highlighted in bold.

All algorithms are implemented in the framework of the PROST planner, so they also use the sound reasonable action pruning and reward lock detection methods that are described by Keller and Eyerich (2012). The PROST planner that was used in IPPC 2011 is equivalent to our UCT base implementation, except that it uses a fixed search depth limitation of 15. Both an unlimited UCT version and PROST are included in our evaluation, the latter being the only algorithm in our comparison that is not anytime optimal. The results impressively reflect the theoretical results: DP-UCT is a clear improvement over the algorithms based on Monte-Carlo backups, clearly indicating that it pays off to directly take probabilities into account. Our main algorithm UCT* outperforms all other algorithms, including PROST, on almost all domains. Its advantages become especially apparent in the domains where DP-UCT does not have an edge over the other algorithms. The improvement in the domains with dense transition matrices, SYSADMIN, GAME OF LIFE, and TRAFFIC, shows that it pays off to search the state space close to the root more thoroughly. Moreover, the result in CROSSING TRAFFIC, where the lookahead that is needed for good policies is among the highest of the IPPC 2011 problems, shows that it also performs well if a higher horizon must be considered. This is a clear sign that the asymmetric form of the search tree with the majority of visits in the promising parts of the search space is preserved.

5 Conclusion

We have presented a novel algorithmic framework, Trial-based Heuristic Tree Search, which subsumes MCTS, Dynamic Programming, and Heuristic Search. We have identified five ingredients that distinguish different algorithms within THTS: heuristic function, backup function, action selection, outcome selection, and trial length. By step-wise combining Monte-Carlo and Full Bellman backups to Partial Bellman backups, we were able to derive the MaxUCT and DP-UCT algorithms. The latter inherits the ability to solve states by considering probabilities from Dynamic Programming, and the use of non-admissible heuristics, the good anytime behavior and the guidance to promising parts of the search space from UCT. Adding the trial length from Heuristic Search led to UCT*, an algorithm that distributes its resources even better in the search space. Our empirical evaluation shows that both DP-UCT and UCT* perform significantly better on the benchmarks of IPPC 2011 than any other considered algorithm, including the winner of IPPC 2011, PROST.

References

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47:235–256, May 2002.

Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to Act Using Real-Time Dynamic Programming. *Artificial Intelligence (AIJ)*, 72(1–2):81–138, 1995.

R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Cameron Browne, Edward J. Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Thomas Keller and Patrick Eyerich. PROST: Probabilistic Planning Based on UCT. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pages 119–127. AAAI Press, June 2012.

Thomas Keller and Malte Helmert. Trial-based Heuristic Tree Search for Finite Horizon MDPs. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, pages 135–143. AAAI Press, 2013.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pages 282–293, 2006.

M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, March 1998.

Discovering Computationally Rational Eye Movements in the Distractor Ratio Task

Xiuli Chen
University of Birmingham
xxc116@cs.bham.ac.uk

Andrew Howes
University of Birmingham
howesa@cs.bham.ac.uk

Richard L. Lewis
University of Michigan
Ann Arbor, USA
rickl@umich.edu

Christopher W. Myers
Air Force Research Laboratory, Performance and Learning Models Branch
Christopher.Myers2@wpafb.af.mil

Joseph W. Houpt
Wright State University
Dayton, Ohio
joseph.houpt@wright.edu

Abstract

In our recent work we have defined reinforcement learning (RL) problems in which the goal is to discover strategies that are computationally rational given a theory of the constraints on human cognition. These strategies are used to predict human behaviours. In this extended abstract we illustrate this use of RL with an example in which distractor ratio phenomena are explained by deriving strategies for eye movements and target detection given constraints on visual acuity. The distractor ratio effect is shown to be a consequence of computationally rational adaptation to the goal of making presence/absence decisions given location noise in the human visual system.

Keywords: Computational rationality, Reinforcement learning, Visual search, Distractor ratio, Bounded optimality

Acknowledgements

This research was supported by a grant awarded from the Air Force Office of Scientific Research to the three authors (AFOSR #12RH05COR) and grants awarded by NASA and the FAA to the second and third authors (NNX12AB08A and E2020272).

1 Introduction

In our recent work we have modeled human visual search by inferring strategies that are computationally rational given a theory of the constraints on human cognition. These strategies are used to predict human behaviour. In this abstract we illustrate these problems with an example in which visual search is explained by deriving strategies for eye movements given constraints on visual acuity.

The starting point for the model is to define a reinforcement learning (RL) problem in which the goal is to discover strategies that are computationally rational given a theory of the constraints on human cognition. In this abstract we illustrate these RL problems with an example in which distractor ratio (DR) phenomena (Bacon & Egeth, 1997; Shen, Reingold & Pomplum, 2000) are explained by deriving strategies for eye movements given constraints on visual acuity (Najemnik & Geisler, 2005). We discuss the role of the computational rationality assumption in guiding the construction of theories of the mechanisms of the mind. In the distractor ratio task the goal is to find a target symbol, for example, a red letter O. The task is more difficult for people, requiring more fixations, when the ratio of the number of distractors with the same color to the number of distractors with the same shape is nearer to 1. For example, in Figure 1 it can be seen that the red letter O can be found quickly in Figure 1A (left panel) and Figure 1C (right panel) but is relatively difficult to find in Figure 1B (middle panel).

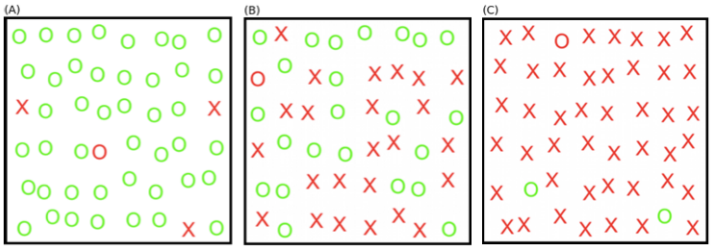


Figure 1: The Distractor-Ratio Task.

One explanation for the DR phenomena is that people move their eyes to salient information (Itti & Kock, 2000). A different explanation, proposed by Myers et al. (2013), is that people find optimal eye movement strategies given constraints on visual acuity. The optimal eye movement strategy for this task makes use of the minority set of distractors to come to a decision about the presence or absence of the target in as few fixations as possible. The minority set, containing the target, is red in Figure 1A and is shape O in Figure 1C. People exhibit a *saccadic bias* in which a higher proportion of saccades are to the minority set than to the majority set. They also exhibit an effect of distractor ratio on number of fixations (Figure 2).

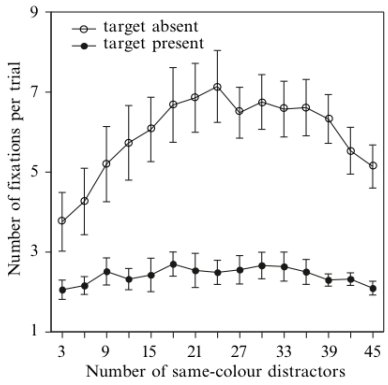


Figure 2: Distractor ratio stimuli when searching for a red O, and results from Shen et al. (2000).

Myers et al. (2013) report a model of a reduced version of the task in which there are only 7 symbols arranged in a single dimensional vector that given the assumption that there is either 1 or 0 targets derives a Bayesian estimate of the probability of each possible display given the available perceptual evidence. Building on Myers et al. model, the model reported in this extended abstract is based on many of the same assumptions but changes are made in an effort to scale to the larger 2 dimensional arrays of symbols that are used in the human experimental task and that are shown in Figure 1.

2 Model

The model selects fixation locations as it searches a 3x3 grid of symbols looking for a target and responds with a target present or target absent decision. The process by which it determines each action is as follows (3 stages):

(1)Percept (foveated vision corrupted by noise with eccentricity from the fovea): Given the current fixation on one of the cells, perception of the content of each cell in the grid generates a noisy representation of the display. This representation is subject to both feature noise and location noise. A consequence of feature noise is uncertainty in whether the content of the cell is red or green, or whether it is a letter O or a letter X. A consequence of location noise is, for example, that a green O that is next to a red O will be more likely to be perceived as red itself than a green O that is on its own. Both feature noise and location noise increase with eccentricity from the fovea. The standard deviation of Gaussian noise at cell i is a function of the eccentricity of i from the foveated cell and a constant such that $\sigma_i = n \times E_i$ where n is the location noise constant. σ_i determines how much the feature at i is affected by neighbouring cells.

(2) State Estimation: Information from the current fixation is integrated with information from previous fixations. The state obtained at each fixation is a 9 element vector each of which is an estimate of the "relevance" of the content of the cell and each of which is updated independently fixation by fixation.

The model adopts a simple feature-vector coding of the display in which each of the 9 locations is represented by 2 real valued features (one for colour and one for shape), where the value 1 is chosen as the target value for each feature, and 0 is the nontarget value. Therefore, the display is represented as two 3×3 matricies, one for color and one for shape. After adding the location and feature noise, there are two 3×3 matrixes that are a hypothesised internal representation of the display. In order to obtain a relevance score for each location the following equation is used to integrate these matricies. For each pair of values at corresponding locations, i.e. V_{color} and V_{shape} we obtain a V for this location of the display.

$$V = \frac{\sigma^2_{shape}}{(\sigma^2_{color} + \sigma^2_{shape})} V_{color} + \frac{\sigma^2_{color}}{(\sigma^2_{color} + \sigma^2_{shape})} V_{shape}$$

where $\sigma_{color}, \sigma_{shape}$ are weights for color and shape respectively. In this preliminary model, the color and shape signal are equally weighted to give an overall relevance score for each location. Thus there is a 3×3 matrix containing the relevance score of each cell. Based on the assumption that image data is captured in parallel from every possible location at each fixation. There is a new noisy observation from each location at each fixation. A Kalman filter is then applied to update each cell given the new observation.

For example, for location i, the state is $x_i(k) = x_i(k-1) + w_i(k-1)$, where k is the time step, the variable $w_i(k-1)$ is the process noise, which is set to 0 as in this version of the task the true state of the display is not dynamic (though in future versions we will explore dynamic displays). For each fixation, there is a new noisy observation at location i. Using the Kalman filter term, the measurement is $z_i(k) = x_i(k) + v_i(k)$, where, $p(v_i) = N(0, R)$ and R is the measurement noise, which is a function of eccentricity of location

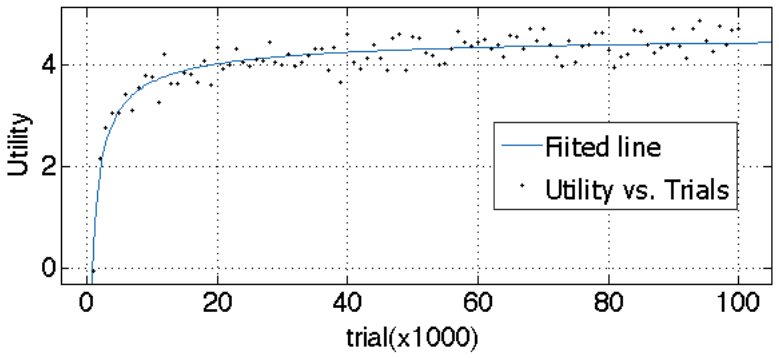


Figure 3: Utility plotted against trial number, demonstrating the extent to which Q-learning generates the computationally rational strategy for the distractor ratio task.

Finally, the model updates the state estimate for each location independently across fixations. At each fixation, after the update of the value of each cell, the fractional part of the scores are rounded to 5 levels, [0 0.2, 0.4 0.6 0.8], to generate a target relevance score vector that then becomes a state. Each state is an entry into a Q-table. The Q-table is a matrix with a row for each state encountered; and a column for each of the actions. This matrix is used to store the state action value function learnt with Q-learning (see below).

(3) Control (Q-learning): At the beginning of the learning, an arbitrary state-action value function was selected. Given the known task, we were able to generate as much experience for the task as the model required. The state-action value function is learned using Q-learning based on the experience generated. The computationally rational strategy is then defined as the strategy that requires the selection of the best action in each state. Specifically, (1) The action space contains the 9 possible next fixation locations, target_present and target_absent. The updated state-action value function is used to select an action using an Epsilon-greedy policy; (2) Given the action selected, the state update process is as described in stage 2 of the model; (3) The reward r, which corresponds to an assumption about the participants' subjective utility function is defined by $r = 10 * C - N$, where $C = 1$ if correct and $C = -1$ if incorrect; N is the number of fixations. Given the experience generated, the model learns the values of the 11 actions for each state that it reaches. These values define the computationally rational strategy. The model was run for 100,000 trials for each noise level n, which was sufficient to obtain asymptotic performance, as shown in Figure 3.

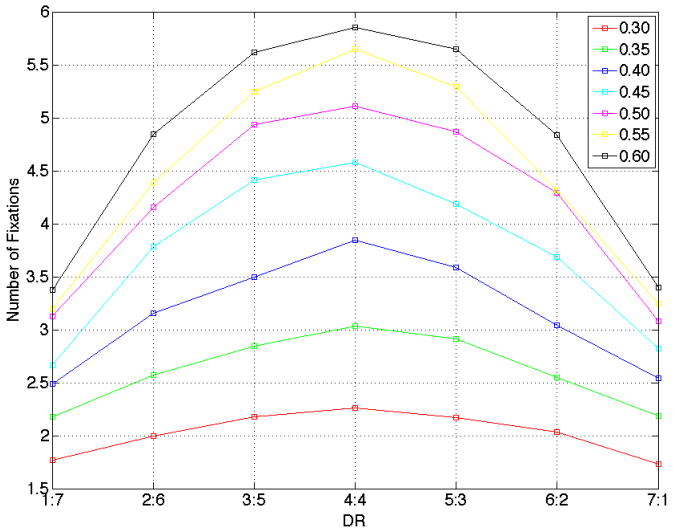


Figure 4: Number of fixations required by the computationally ration strategy for each level of distractor ratio (DR) and each level of location noise.

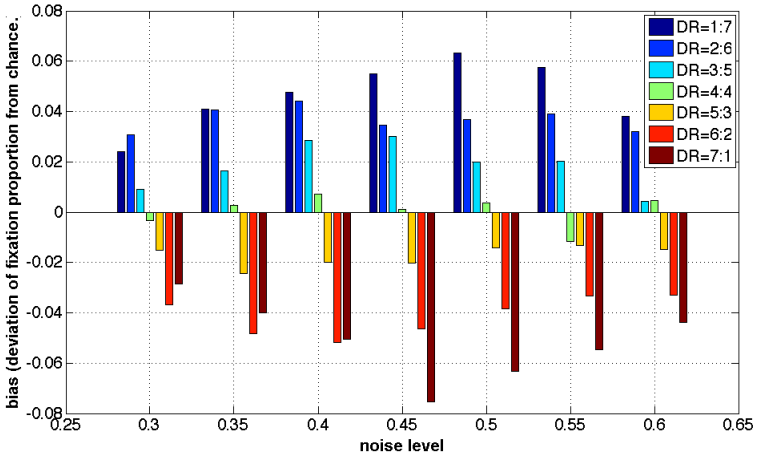


Figure 5: Saccadic bias against level of location noise in visual perception

3 Results

The search efficiency for the computationally rational strategy at each level of location noise is shown in Figure 4. This shows that the model generates the same shape distractor ratio effect as in humans (Figure 2), where more fixations are

required for ratios close to 1. The effect of location noise on saccadic bias is illustrated in Figure 5. The highest saccadic bias occurs with middle levels of noise. In contrast, if there is no, or little noise, then there is reduced saccadic bias in the rational strategy. This is because in the absence of location noise the target can be fixated immediately. Similarly at high noise levels there is reduced saccadic bias but here it is because so little information is perceived that the fixation pattern is at chance. This pattern is accentuated at more extreme distractor ratios. These two results extend the findings of Myers et al. (2013) to show that the distractor ratio effect can be explained as a side-effect of computationally rational strategies adapted to location noise that increases with eccentricity from the fixation. The new model also extends the Myers et al. model by generating centre of gravity effects (Najemnik & Geisler, 2005) and speed/accuracy trade-offs (not illustrated here).

4 Discussion

The model reported here is an extension of a computationally rational account of the distractor ratio effect in visual search that was first reported by Myers, Lewis and Howes (2013). Rather than explaining the effect as a consequence of salience (Itti and Kock, 2000), the rational model explains it as an adaptation to information processing constraints, specifically location noise, which is inherent to the human visual system. The extensions reported in the current abstract define the eye movement problem given noisy perception and learn the computationally rational strategy for eye movements and present/absent decision. The strategy is optimal given the reward function and the limitations imposed by the model of visual perception, which is subject to both feature and location noise.

In this work Q-learning was used as an analytic tool for determining the optimal policy given a theory of a limited perceptual capacity. This role for an RL algorithm has previously been suggested by Chater (2009) who contrasted rational uses of definitions of the RL problem to the statement of RL mechanisms as theories of human and animal learning. In the rational approach it is the theory of resource limits, here limits on visual information processing, that is under test, not a theory of learning. The discovery of a computationally rational strategies is critical to explaining human information processing for two reasons, (1) the distractor ratio phenomena are shown to be a consequence of these strategies and not others (Myers et al., 2013), (2) the problem of fitting overly flexible models to the data is avoided as the analysis maximizes utility rather than maximizing fit (Howes, Lewis & Vera, 2009) thereby supporting the reverse engineering of the mind (Lewis, Howes & Singh, in press; Dayan, in press; Howes, Lewis & Vera, 2009).

There is a substantial amount of work to be done to test and develop the reported model. For a start it needs to be scaled to the 48 element displays that were used in the original experiments reported by Shen, Reingold and Pomplum (2000). Target present data and interactions also needs to be modelled. We anticipate that this scaling is tractable using the display sampling, noisy perception, gridding, and Q-learning algorithm described in this abstract. In addition, we need to make a fuller comparison to the Bayesian model reported by Myers et al. (2013). Comparison to the performance of the Bayesian model will help expose the relevant properties of both. In addition, we wish to address how, or whether, people adapt to the distractor ratio task with little experience of the local task.

5 References

Bacon, W., Egeth, H., 1997. Goal-directed guidance of attention: Evidence from conjunctive visual search. J. Exp. Psychol.: Hum. Percept. Perform. 23, 948–961.

Chater, N. (2009). Rational and Mechanistic Perspectives on Reinforcement Learning. Cognition, 113(3), 350-364.

Dayan, P. (in press). Rationalisable irrationalities of choice. Topics in Cognitive Science.

Howes, A., Lewis, R. L., & Vera, A. (2009). Rational adaptation under task and processing constraints: Implications for testing theories of cognition and action. *Psychological review, 116*(4), 717-751.

Itti, L., & Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10-12), 1489–1506.

Lewis, R.L., Howes, A. & Singh, S. (in press). Computational Rationality: Linking Mechanism and Behavior through Bounded Utility Maximization. *Topics in Cognitive Science*.

Myers, C.W., Lewis, R.L. & Howes, A. (2013). Bounded optimal state estimation and control in visual search: explaining distractor ratio effects. In M. Knauff, M. Pauen, N. Sebanz, & I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society* (pp. TBA). Austin, TX: Cognitive Science Society.

Najemnik, J., & Geisler, W. S. (2005). Optimal eye movement strategies in visual search. *Nature*, 434(7031), 387-391.

Online Value Function Improvement

Mitchell Keith Bloch
University of Michigan
2260 Hayward Street
Ann Arbor, MI. 48109-2121
bazald@umich.edu

John Edwin Laird
University of Michigan
2260 Hayward Street
Ann Arbor, MI. 48109-2121
laird@umich.edu

Abstract

Our goal is to develop broadly competent agents that can dynamically construct an appropriate value function for tasks with large state spaces so that they can effectively and efficiently learn using reinforcement learning. We study the case where an agent's state is determined by a small number of continuous dimensions, so that the problem of determining the relevant features corresponds roughly to that of determining the appropriate level of discretization of the continuous values. We adopt hierarchical tile coding, which applies state aggregation at multiple levels of state abstraction simultaneously. Using our formulation, it is possible to capture the advantages of learning with state abstractions ranging from general to specific using linear function approximation. We then develop a novel algorithm for incrementally refining the degree of state abstraction, based on cumulative absolute temporal difference error, which produces a sparse non-uniform tile coding. We empirically evaluate our approach in the Puddle World and Mountain Car environments. The results demonstrate that the static and incremental hierarchical tile codings significantly outperform individual tilings and multilevel tile codings (CMACs) for initial learning. Our results also indicate that the incrementally constructed tilings perform nearly as well as the full hierarchical tile coding while requiring an order of magnitude fewer weights.

Keywords: Reinforcement Learning, Online Learning, Incremental Learning, Value Function, Tile Coding

Acknowledgements

We acknowledge the funding support of the Air Force Office of Scientific Research, contract FA2386-10-1-4127.

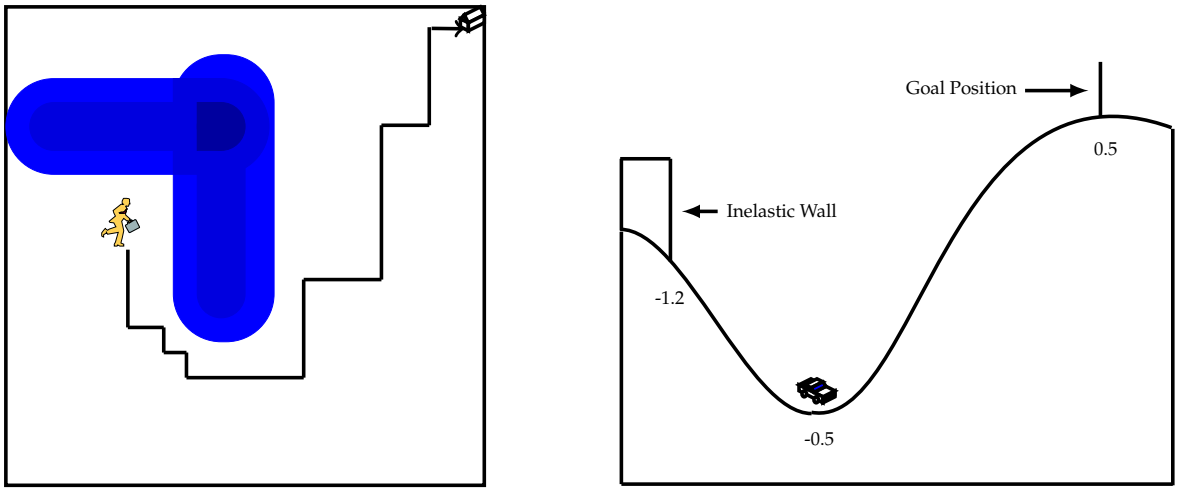
1 Introduction

At a broad level, our goal is to build agents which can perform difficult tasks in environments with large state-spaces that are described by a large number of features, some of which may be continuous. In this work, we focus on continuous features and explore a novel strategy for determining when to refine a tile coding[1] in order to allow an agent to improve its policy. We then demonstrate that a tile coding consisting of multiple fixed tilings of variable resolution can do significantly better than any single tiling. We develop incremental hierarchical tile codings that can do nearly as well as static hierarchical tile codings, while using significantly less memory.

2 Environments

We experiment with Puddle World and Mountain Car—two environments with infinitely large state-spaces due to their continuous features. These problems present the difficulty that different parts of their state-spaces warrant reasoning at different levels of precision.

Figure 1(a) shows an example of Puddle World, where an agent can move North, South, East, or West, and where the goal is for an agent to move from an initial location to a goal region. [Sutton, 1996] The world contains "puddles" that are capsule shaped regions whose depth increases from their edges to their centers. The environment is fully observable, but the agent's steps are stochastic, resulting in step sizes between 0.04 and 0.06 units. As the x and y positions are real-valued, the state-space is infinitely divisible, and therefore not perfectly discretizable. The agent receives a penalty of -1 for each step and an additional penalty proportional to the depth of each puddle at its current location.



(a) Puddle World. (b) Mountain Car.

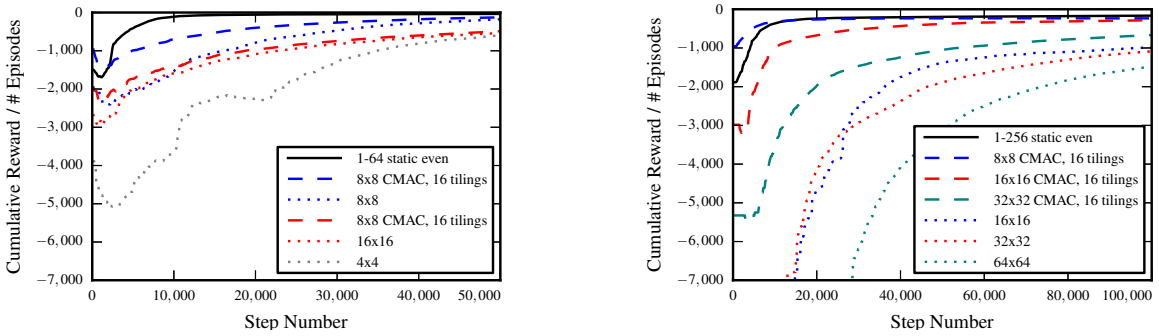
Figure 1: Depictions of our environments.

Figure 1(b) shows the canonical Mountain Car [Singh *et al.*, 1996], where an agent can control the car's motor (left, idle, right), and where the goal is for the agent to move the car from the basin, at rest, to the top of the mountain on the right. Given gravity $-0.0025\cos(3x)$, and a car with power 0.001, the car is incapable of climbing the mountain starting from rest. It is essential to build up potential energy by backing up the hill on the left before moving to the right. The agent receives a penalty of -1 for each step.

3 Static Hierarchical Tile Coding

We examine hierarchical tile codings where there can be multiple non-overlapping tilings at different resolutions, such as 2x2, 4x4, 8x8, ..., the goal being to support learning at different levels of abstraction. In strict hierarchical tile coding, there are n levels of tilings, with separate tile codings for each action. This introduces a credit assignment problem for hierarchical tile codings, which we solve with an even credit assignment strategy, as is typical for linear function

[1] A tile coding or CMAC (Cerebellar Model Articulation Controller) consists of one or more tilings that partitions a continuous space into a fixed number of a regions, each corresponding to a binary feature. For a detailed explanation, see section 8.3.2 of Sutton and Barto [1998].



(a) Puddle World. (b) Mountain Car.

Figure 2: Performance for several agents using single tilings, traditional CMACs, and a static hierarchical tile coding.

approximation. General tilings (such as 2x2) receive updates frequently, allowing them to converge quickly. They act as a baseline to speed learning in the more specific tilings (such as 64x64).

We explored hierarchical tile codings with varying subsets of the tilings, 1x1-64x64, such as omitting the most specific or the most general tilings, and none achieved better performance than using the complete hierarchy. Zheng *et al.* [2006], Grzes and Kudenko [2008], and Grzes [2010] previously explored using only two tilings with varying state abstraction.

In our experiments we use an epsilon-greedy exploration strategy ($\varepsilon = 0.1$ for Puddle World and $\varepsilon = 0.01$ for Mountain Car) with a random tiebreak, Q-learning with a learning rate of 0.1 in Puddle World and 1.0 in Mountain Car, and a discount rate of 0.999, and we initialize weights to 0. Figure 2(a) shows the results of using specific tilings for Puddle World, together with traditional CMACs (consisting of multiple identical tilings with different offsets), and hierarchical tile codings. We compare against not only the best CMACs, but also the CMACs corresponding to our single tilings, in order to demonstrate the performance degradation that still occurs as the tile sizes decrease. The y-axis shows cumulative reward per episode, with the x-axis showing total steps. Each data point is an average of 20 runs. We ran experiments with agents using 32x32 and 64x64 tilings; however, they did not start to converge until $> 50,000$ steps and so are not included in the figure. Figure 2(b) shows corresponding results for Mountain Car.

The most dramatic feature of the figure is that the hierarchical tile coding does significantly better than any individual tiling. One hypothesis that these results dispel is that the advantage of the hierarchy is just in hedging the bet as to which tiling is best. Instead, it does much better than even the best single tiling (8x8). The hypothesis it supports is that it can take advantage of the fast learning possible with the more general tilings because of their more frequent updates, while taking advantage of the accuracy provided by the more specific tilings. Moreover, the more specific tilings (such as 32x32 and 64x64) do not drag down the rate of learning.

Additionally, hierarchical tile coding performs better than CMACs for the vast majority of CMAC parameter settings we tried. The hierarchical tile codings dominate all of the CMACs significantly in Puddle World, and most CMACs significantly in Mountain Car. The 8x8 CMAC with 16 tilings does nearly as well in Mountain Car, but only achieves performance comparable to a 1-64 hierarchical tile coding, and a parameter sweep was required to discover it.

So why does hierarchical tile coding work so well? One important feature of both of these environments is that there is continuity in the mapping from the feature space to the weights or Q-values—entries in the value function that are near each other spatially tend to have similar values.

4 Dynamically Refining the Value Function

Although hierarchical tile coding is very effective for these domains, there are two significant problems. First, it requires committing to a set of tilings from the beginning of the task. This is not always be possible for arbitrary tasks. The second problem is that it requires large memories to hold weights for every tiling of the hierarchy, growing exponentially with each additional tiling of the hierarchy. In the future, we plan to study domains with more features where it will be impossible to store weights for every tiling. Thus, we want to develop approaches where the agent does not need to commit to a specific level of tiling, and where the number of weights that must be maintained is minimized.

Incremental approaches to expanding the hierarchy have the potential of satisfying both of these criteria. In incremental approaches, the agent starts with only the most general tilings, which in this case include both a 1x1 and a 2x2 discretization of the task features. Based on experience, the agent determines when it might be useful to have a more specific state abstraction (or more refined discretization) of one of the tiles. An additional level of tiling is created that covers just the

chosen tile. Those new tiles are initialized to 0, and with experience, they are updated to match the differences at their level of detail. Additional tiles are expanded, leading to a non-uniform tiling of the space.

The approach explored by Geramifard *et al.* [2011] tracks a fringe of feature conjunctions and expands the set of weights over time. It makes decisions about whether to refine the value function using a static criterion based on TD (temporal difference) error in the fringe, rather than a globally relative criterion like that employed by Munos and Moore [1999] and ours. Additionally it is uses much more memory than our approach given the use of a fringe, and because it may track weights for the full power set of features, rather than using an approach based on a decision tree. For that reason, our approach better satisfies our efficiency criteria.

Inspired by *Stdev_Inf* [Munos and Moore, 1999], the metric we choose to encompass these properties is cumulative absolute temporal difference error (CATDE). TD error—the delta essential to temporal difference methods—is highest in regions of high variance. Tracking the CATDE for a tile in parallel with each weight provides a metric which increases more quickly when the variance associated with taking an action is high, and when that action is taken frequently.[2] Environmental stochasticity artificially inflates the CATDE values but, because the variance is estimated locally, the impact on our metric is less than the impact on *Stdev_Inf*. Additionally, CATDE can be tracked incrementally, requiring low computational costs.

Given the CATDE metric, we choose to select the tiles with the greatest CATDE for refinement. In order to make this decision procedure incremental, it is essential to have an efficient algorithm for tracking the mean and variance for CATDE throughout our value functions. Incrementally tracking a mean for a set of values is fairly trivial, but incrementally calculating the variance requires a modified version of an algorithm provided by Knuth and Welford [Knuth, 1997; Welford, 1962]. We implemented additional methods to allow updating values and removing values from the set.

Whenever a tile is refined, the CATDE for its region is reset to 0. Initially, the CATDE metric will result in significantly faster refinement to some regions than others. But as time goes to infinity, tiles will tend to split at the same rate, as regions which receive greater refinement will accumulate TD error over smaller subsets of the state-space over time.

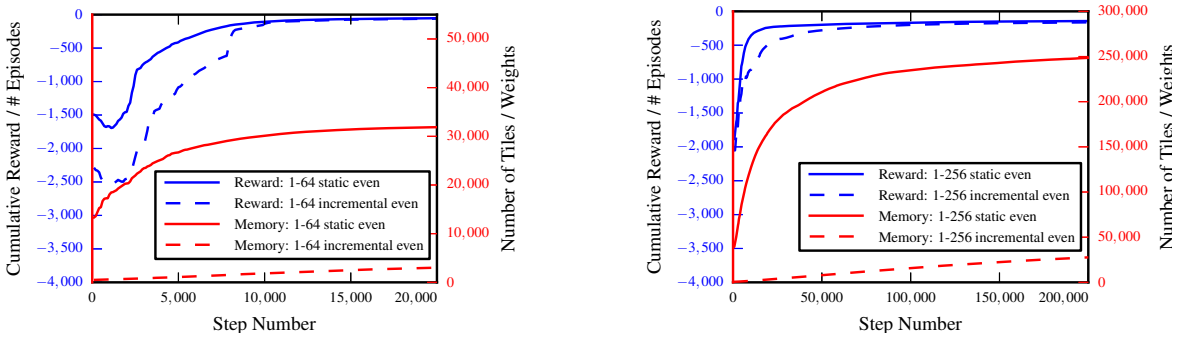
Given a tile with CATDE $= c$, our mean estimate, μ, and our variance estimate, σ^2, we refine a tile if

$$c > \mu + z\sigma^2 | z = 0.5 \tag{1}$$

and the tile has not been visited in the past 20 steps. z is chosen to determine how selective the agent should be in choosing which tiles to refine. The 20 step threshold prevents overzealous refinement.

Given the limited number of features in both Puddle World and Mountain Car, our agents simply alternate back and forth between the dimensions when making these refinements.

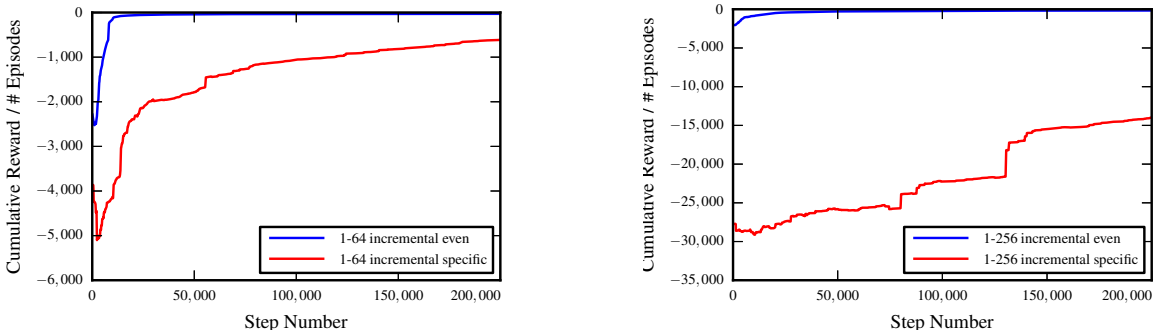
Figure 3(a) shows data for the Puddle World domain. Data for static hierarchies, with tilings of resolutions 1x1 through 64x64, and for incremental hierarchies, with tilings of resolutions initially between 1x1 through 2x2 are presented together. The number of weights is overlaid over the performance data so that one may compare their performance to their memory efficiency as time passes. The cumulative performance of the incremental hierarchies is within 13% of the static hierarchies by 20,000 steps, however the number of weights is reduced by 90%. Figure 3(b) shows corresponding data for the Mountain Car domain. Data for static hierarchies, with tilings of resolutions 1x1 through 256x256, and for incremental hierarchies, with tilings of resolutions initially between 1x1 through 2x2 are presented together. The cumulative



(a) Puddle World. (b) Mountain Car.

Figure 3: Performance and memory usage of an agent using a static hierarchical tile coding and another agent using an incremental hierarchical tile coding.

[2]It is critical that CATDE is cumulative. If it were non-cumulative, it would essentially eliminate the idea of influence from *Stdev_Inf*, and turn it into a kind of variance metric. This has been confirmed empirically (not shown).



(a) Puddle World. (b) Mountain Car.

Figure 4: Performance of agents using incremental hierarchical tile codings with different credit assignment strategies.

performance of the incremental hierarchies is within 14% of the static hierarchies by 200,000 steps, however the number of weights is reduced by 89%.

Figure 4 contrasts our incremental hierarchical tile coding against a tile coding which gives all credit to the most specific tiles. This is equivalent to comparing against an adaptive tile coding which splits tiles, keeping only one tile per region, as described by Munos and Moore [1999] and Whiteson *et al.* [2007]. The performance of the even credit assignment strategy dominates the performance of the specific credit assignment strategy. The results of these experiments are presented in figure 4 with a significant change in scale for both the x and y-axes.

5 Conclusion

In summary: we demonstrated that static hierarchical tile codings dominate individual tile codings and CMACs in two domains; we developed an incremental hierarchical tile coding which performs well while saving memory; and we demonstrate that incremental hierarchical tile codings dominate incrementally split, non-hierarchical tile codings.

References

[Geramifard *et al.*, 2011] Alborz Geramifard, Finale Doshi, Josh Redding, Nicholas Roy, and Jonathan P. How. Online discovery of feature dependencies. In Lise Getoor and Tobias Scheffer, editors, *ICML*, pages 881–888. Omnipress, 2011.

[Grzes and Kudenko, 2008] Marek Grzes and Daniel Kudenko. Multigrid reinforcement learning with reward shaping. In *ICANN (1)*, pages 357–366, 2008.

[Grzes, 2010] M. Grzes. *Improving exploration in reinforcement learning through domain knowledge and parameter analysis*. PhD thesis, University of York, 2010.

[Knuth, 1997] Donald E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*. Addison-Wesley Professional, 3 edition, November 1997.

[Munos and Moore, 1999] Remi Munos and Andrew Moore. Variable resolution discretization in optimal control. *Machine Learning Journal*, 1999.

[Singh *et al.*, 1996] Satinder Singh, Richard S. Sutton, and P. Kaelbling. Reinforcement learning with replacing eligibility traces. In *Machine Learning*, pages 123–158, 1996.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. Reinforcement learning i: Introduction, 1998.

[Sutton, 1996] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

[Welford, 1962] B. P. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3):419–420, 1962.

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

[Zheng *et al.*, 2006] Yu Zheng, Siwei Luo, and Ziang Lv. Control double inverted pendulum by reinforcement learning with double cmac network. *Pattern Recognition, International Conference on*, 4:639–642, 2006.

Solving for Best Responses in Extensive-Form Games using Reinforcement Learning Methods

Amy Greenwald, Jiacui Li, Eric Sodomka, Michael Littman
Department of Computer Science
Brown University
Providence, RI 02912
{amy, jl52, sodomka, mlittman}@cs.brown.edu

Abstract

We present a framework to solve for best responses in extensive-form games (EFGs) with imperfect information by transforming the games into Information-Set MDPs (ISMDPs), and then applying simulation-based reinforcement learning methods to the ISMDPs. We first show that, from the point of view of a single player, an EFG can be represented as an Information-Set POMDP (ISPOMDP) whose states correspond to the nodes in the EFG. This ISPOMDP can then be further represented as an ISMDP, whose states correspond to the information sets in the EFG. Because the transformations are lossless, every optimal policy in the ISMDP is a best response in the original EFG.

Our approach to finding a best response in an EFG, therefore, is to first apply the aforementioned transformations, and to then use simulation to learn the ensuing ISMDP and standard techniques (e.g., dynamic programming) to solve it. There are two challenges to effectively learning the ISMDP through simulation: the ISMDP state space is exponential in the horizon, and we cannot resample actions during simulation. We prove that simulation can still be guaranteed to learn near-optimal best responses with high probability, although the sample complexity depends explicitly on the size of the state space. Using our best-response finding algorithm as a subroutine, we further develop two algorithms, one that implements approximate best-reply learning dynamics, and another that approximates ϵ-factors of strategy profiles in EFGs. We evaluated these algorithms by applying them to several sequential auction domains.

Keywords: Game Theory, Model-Based Reinforcement Learning, Partially Observable Markov Decision Process

1 Introduction

Real-world interactions between individuals and organizations are commonly modeled as extensive-form games (EFGs), in which players act in sequence, and payoffs to one player depend on the actions of all players. Each player in an EFG is predicted to play her *best response*, which is a strategy that maximizes her expected payoff against opponent strategies. This notion of best response is central to game theory. It forms the basis of Nash Equilibrium (NE), the canonical game-theoretic solution concept: at NE, all players simultaneously play best responses to one another [12]. Best responses also underpin classic game-theoretic learning schemes, such as *best-reply dynamics* and *fictitious play* [1].

In this paper, we develop an RL-based algorithm that approximates best responses in EFGs of imperfect information. Given opponent strategies, an EFG from one player's perspective can be represented as an Information-Set POMDP (ISPOMDP), which can be further represented as an Information-Set MDP (ISMDP). The ISMDP can be learned via simulation, after which an optimal policy can be approximated using standard techniques (e.g., dynamic programming). Because the transformation from EFG to ISMDP preserves all strategically relevant components, this optimal ISMDP policy is a best response in the original EFG. Although the seemingly similar approach of POMCP (Partially Observable Monte Carlo Planning) also applies [16], it is not as efficient in the specific domain we address. In extensive form games with incomplete information, each player receives a private valuation drawn from a distribution before each play of the game. As long as valuations are independent across players[1], this private information only affects that player's payoff but not the dynamics of the opponents she face. Our approach exploits this independence and efficiently learn opponent dynamics (encapsulated in an ISMDP) separately, while UCT-based methods such as POMCP learn a best policy for each possible valuation so is more computationally intensive.

2 Transforming EFGs into ISMDPs

In this section we show how to transform EFGs to ISMDPs and illustrate the procedure with an example in Figure 1.

EFG An EFG is specified by a rooted game tree with labeled nodes and action branches [5], as illustrated in Figure 1(a). Each play of the game is a trajectory from the root to some leaf, and the payoffs of each player depend on which terminal node the game ended on. Each non-terminal node is a decision node controlled by one player who chooses from the available action branches. Therefore, the game trajectory is jointly determined by all players. To model uncertainty from the environment, there is a special player *nature* who chooses actions at *chance nodes* with probabilities commonly known to all players. In Figure 1(a), each decision node is labeled with the identity of its controlling player, while chance nodes are labeled 0.

Real-world situations are often modeled as EFGs with *imperfect information* in which each player's decision nodes are partitioned into *information sets*, and the player only observes which information set she is in when choosing actions. She does not know the exact node and therefore has to make decisions under uncertainty. In Figure 1(a), nodes in the same information set are circled together with dashed lines, and a node that isn't circled belongs to a singleton information set. For example, player 1 does not observe nature's move before her first decision, but player 2 does. Moreover, player 2 knows which action player 1 has taken if nature chose the right action but not otherwise.

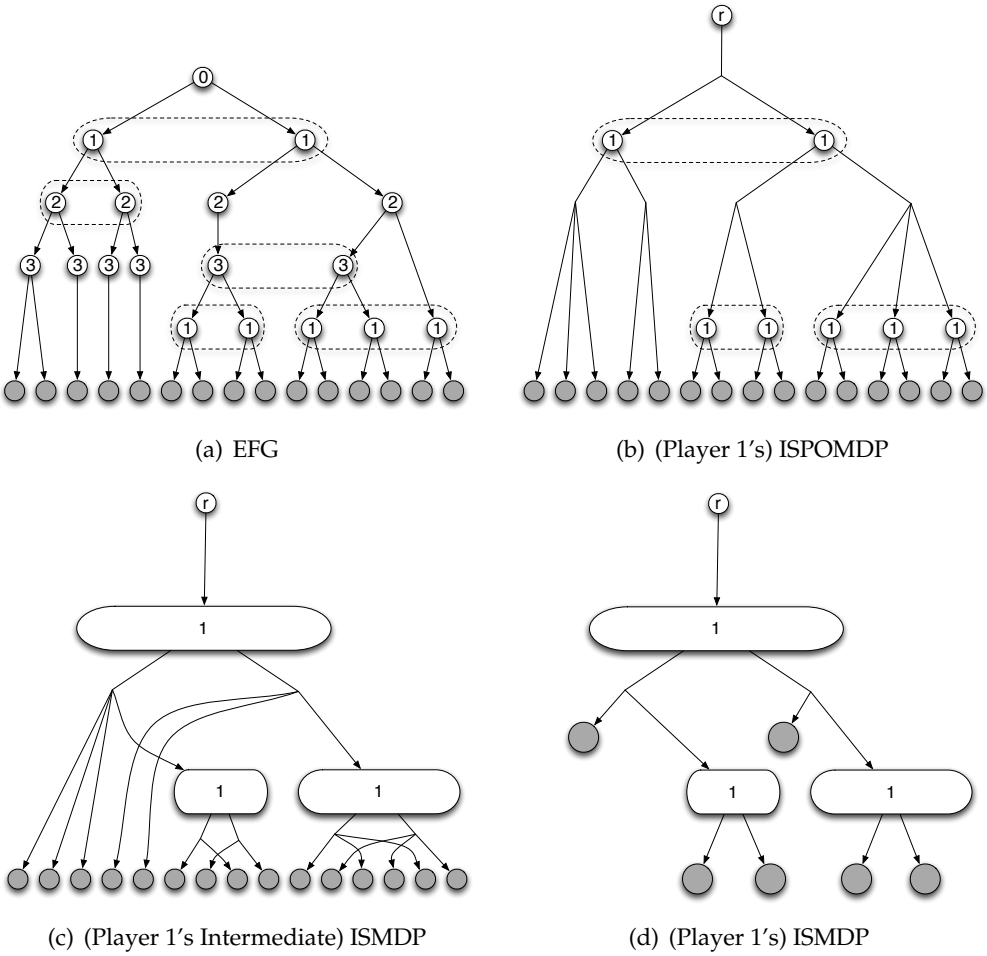
Each player's strategy is a mapping from her information sets to actions, possibly with randomizations over actions. Modeled as expected payoff maximizers, each player wants to play her *best response*, which is a strategy that maximizes her expected payoff given opponent strategies. When knowing opponent strategies, the problem of finding a best response reduces to a single-agent optimization problem under uncertainty [13]. Therefore from one player's perspective, EFGs can be transformed into POMDPs , which we call Information-Set POMDPs (ISPOMDPs).

EFG to ISPOMDP The hidden states in the ISPOMDP correspond to the controlling player's decision nodes and terminal nodes in the EFG. The uncertainty due to opponent and nature actions in EFGs are incorporated in the probabilistic transitions between hidden states in the ISPOMDP. The terminal hidden states are associated with rewards which correspond to the EFG rewards in terminal game nodes. At any non-terminal hidden state, the player receives an observation of which information set—a set of hidden states—she is in. Figure 1(b) shows an example ISPOMDP for player 1.

ISPOMDPs are special cases of POMDPs with particular structure inherited from the EFGs. For example, while general POMDPs can be connected graphs, ISPOMDPs are always trees because EFGs are trees. The most crucial feature is that each observation in the ISPOMDP tells the player which set of hidden states she can be in. The belief state associated with an observation has positive probabilities over hidden states in that information set and zeros probabilities elsewhere.

[1] This is a common assumption in games, .e.g, the independent private values assumption in auction games. Our algorithm requires this assumption to apply.

ISPOMDP to ISMDP We can further transform the ISPOMDP into an Information-Set MDP (ISMDP). An example appears in Figure 1(d), with Figure 1(c) representing an intermediate step. The non-terminal decision states in the ISMDP correspond directly to information sets in the EFG. The terminal states are groups of hidden terminal states. For example, we can group together the three leftmost hidden terminal states in Figure 1(c), where the reward of the resulting state is a probabilistic mixture of the rewards from the hidden states. This reduces the ISMDP state space. The transition probabilities between ISMDP states are defined as transition probabilities between corresponding belief states, which can be derived with Bayesian updating. The ISMDP policies are mappings from states to actions.



(a) EFG (b) (Player 1's) ISPOMDP

(c) (Player 1's Intermediate) ISMDP (d) (Player 1's) ISMDP

Figure 1: An example of transforming an (a) EFG first to (b) ISPOMDP and then to (d) ISMDP from one player's perspective.

We argue the transformation from EFG to ISMDP is *lossless* from the perspective of best response computation, because all strategically relevant components have been preserved. For example, the player knows which ISMDP state she is in when making decisions, just like she knows which information set she is in when playing in the EFG. The knowledge about opponent strategies in the EFG is incorporated into the transition probabilities between states in the ISMDP. The expected payoff in terminal states of the ISMDP is derived directly from the payoffs in the EFG. Therefore the following result follows.

Theorem 1. *Consider an EFG that is transformed into an ISMDP from one player's perspective using the procedure described above. Then a policy / strategy is optimal in the ISMDP if and only if it is a best response in the EFG.*

3 Learning and Solving ISMDPs by Simulation

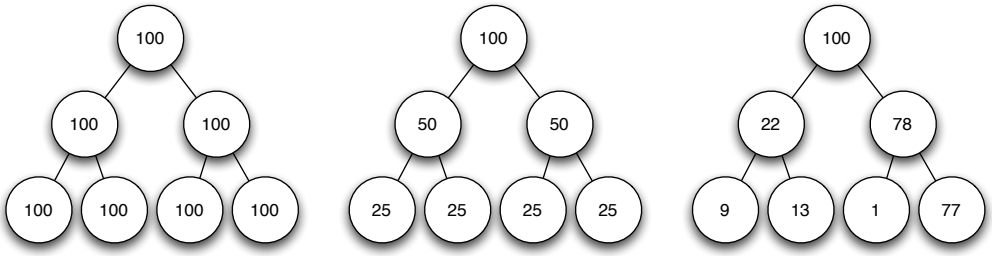
Once an EFG is transformed into an ISMDP, we can leverage existing reinforcement learning algorithms for learning and planning. Our algorithm learns an approximate ISMDP by simulation, and then uses dynamic programming to solve for an optimal policy, which is an approximate solution to the true ISMDP. The simulation of player i's ISMDP is best understood as repeatedly replaying the EFG. We sample nature's moves and simulate opponents choosing actions

according to their respective strategies, while player i explores all actions probabilistically. Recall that each information set in the EFG uniquely corresponds to a state in the ISMDP. After collecting histories of many game trajectories, the transition probabilities $P(s'|s,a)$ are approximated by the empirical probability of s' when i chooses a at s. Similarly, the rewards $R(s,a)$ are approximated using sample averages of rewards experienced.

Properties inherited from the original EFG structure, however, pose important challenges to learning the ISMDP effectively. First, the number of states is exponential in the horizon. Since the player remembers her own past moves, even without observation of opponent and nature actions, the number of information sets after H actions is at least A^H in the EFG, where A is the number of actions to choose from. This means there are at least A^H states in the ISMDP because they correspond directly to information sets. As we show in Theorem 3, the number of samples required for learning a near-optimal response has a lower bound that depends on A^H.

Secondly, each simulation generates a single uninterrupted trajectory from root to some leaf in the ISMDP, so the sample points get divided up sparsely deeper into the ISMDP. Moreover the sample points can be distributed unevenly among states in the same level, as the number of times an ISMDP state is reached depends on probabilistic moves by nature and other-player strategies. Therefore even after numerous simulations, some states can still be rarely visited and their dynamics poorly learned, as illustrated in Figure 2(c).

Kearns et al. [8] designed a Sparse Sampling algorithm (SS) to efficiently plan in large MDPs with exponential state space, but it requires the ability to resample actions any point along a trajectory, a setting in which the simulator is able to make the distribution of sample points look like that in Figure 2(a). This is not possible in our setting: part of the uncertainty in $s' \sim P(\cdot|s,a)$ depends on the earlier actions of nature and opponents in the game trajectory, so resampling s' independently requires resampling all those actions. But then the game could have taken a different route without passing s.



(a) (b) (c)

Figure 2: The numbers represent the number of times each state is reached in simulation. (a): The case assumed achievable in Sparse Sampling of [8]. (b): The best case in our setting. (c): A case that can occur in our setting. The state is sampled only once is learned poorly even if the total number of simulations is large.

These two challenges - exponential state space and the fact that some states will always be poorly learned - make it extremely difficult to derive optimality-guarantees of the learned policy. Despite the difficulties, we prove a baseline upper bound on the samples required for learning a near-optimal policy. The proof depends on the realization, as pointed out by Kearns and Singh [7], that states rarely reached in the simulation contribute less to the overall payoff precisely because they are so unlikely to be reached. The full proof is available in a technical report.

Theorem 2. *Suppose the simulator explores the ISMDP with* balanced wandering*: at each state, choose the least chosen action and break ties uniformly randomly. Consider any A-action ISMDP tree with horizon H, rewards bounded in $[0,1]$, and branching factor S (therefore the number of states in level h is S^h). For any $\epsilon, \delta > 0$, to ensure the learned policy is ϵ-optimal with probability at least $1-\delta$, it suffices to have the following sample size:*

$$N = \mathcal{O}\left(\left(\frac{2ASH}{\epsilon}\right)^{2H}\left(\log\frac{1}{\delta} + H\log S\right)\right) \tag{1}$$

This bound can perhaps be improved upon by considering selective sampling schemes, such as UCT [9]. We also prove a lower bound which says it is impossible to get rid of an exponential dependence on the horizon H:

Theorem 3. *For any simulation scheme, possibly adaptive, there exists an ISMDP such that the algorithm must generate at least $\Omega\left(\frac{A^H}{\epsilon}\log\frac{1}{\delta}\right)$ samples to guarantee the learned policy be ϵ-optimal with probability $1-\delta$.*

Implementation To evaluate the performance of our ISMDP-based approach, we ran sequential auction experiments represented as EFGs with imperfect information. A complete description of these experiments can be found in Greenwald et al. [3]. Starting from three sequential auction models with known (analytically-derived) equilibria, we trans-

formed them into ISMDPs and computed best responses to equilibrium strategies. Our algorithm not only found close approximation of the known best responses, but also discovered new heretofore unknown ones.

We further implemented two algorithms that call our best-response algorithm as a subroutine. The first approximates best-reply dynamics [1]. We used this algorithm to find near-equilibria in sequential auctions with state spaces larger than those previous solved in the literature. The second algorithm approximates the ϵ-factor, a measure of stability of near-equilibrium strategy profiles in games. We used this ϵ-factor approximation algorithm to conclude that our best-reply dynamics converged to stable near-equilibria.

4 Related Literature and Future Work

In terms of applying reinforcement learning methods to games, many variants of Q-learning [11][6][4] have been applied to approximate game theoretic equilibrium. While we consider EFGs, they applied their methods to stochastic games. In the auction domain, the iterative method *self-confirming price prediction* employed in [14] is very similar to the best response dynamics we implemented. Our method of epsilon-factor approximation is similar to the *Ex-post check* in [2]. Sampling complexity bounds for reinforcement learning have been studied by many; a summary appears in [17].

This paper focuses on the theoretical framework of converting EFGs to ISMDPs, and the next step is to leverage reinforcement learning methods to scale up the algorithm to large games. The exponential state space in ISMDP and the inability to resample actions in simulation will remain the main challenges. Our current algorithm is a model-based reinforcement learning method, but model-free ones should also be considered. Selective sampling schemes like UCT [9], as well as dimensionality reduction methods like state-space aggregation [10] and value function approximation [15] will be explored in future work.

REFERENCES

[1] D. Fudenberg and D.K. Levine. *The theory of learning in games*, volume 2. MIT press, 1998.
[2] Sam Ganzfried and Tuomas Sandholm. Computing equilibria in multiplayer stochastic games of imperfect information. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 140–146, 2009.
[3] Amy Greenwald, Jiacui Li, and Eric Sodomka. Approximating equilibria in sequential auctions with incomplete information and multi-unit demand. In *Advances in Neural Information Processing Systems 25*, pages 2330–2338, 2012.
[4] Amy Greenwald, Martin Zinkevich, and Pack Kaelbling. Correlated q-learning. In *In Proceedings of the Twentieth International Conference on Machine Learning*, 2003.
[5] Sergiu Hart. Games in Extensive and Strategic Forms. In *Handbook of Game Theory*, volume 1. 1992.
[6] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003.
[7] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49.
[8] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.
[9] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.
[10] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.
[11] Michael L Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328. Morgan Kaufmann Publishers Inc., 2001.
[12] John Nash. Non-Cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951.
[13] A. Neyman. From markov chains to stochastic games. *Stochastic Games and Applications*, 570:397–415, 2003.
[14] Anna Osepayshvili, Michael P Wellman, Daniel Reeves, and Jeffrey K MacKie-Mason. Self-confirming price prediction for bidding in simultaneous ascending auctions. *arXiv preprint arXiv:1207.1400*, 2012.
[15] Ronald Parr, Christopher Painter-Wakefield, Lihong Li, and Michael Littman. Analyzing feature generation for value-function approximation. In *Proceedings of the 24th international conference on Machine learning*, pages 737–744. ACM, 2007.
[16] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010.
[17] Alexander L Strehl, Lihong Li, and Michael L Littman. Reinforcement learning in finite mdps: Pac analysis. *The Journal of Machine Learning Research*, 10:2413–2444, 2009.
[18] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

Relative Bellman Error: An Offline Evaluation Metric for Comparing Value Functions

Vukosi Marivate
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
vukosi@cs.rutgers.edu

Michael Littman
Department of Computer Science
Brown University
Providence, RI 02912-1910
mlittman@cs.brown.edu

Abstract

Reinforcement learning (RL) algorithms are typically evaluated online—a value function or policy is used to control the target system and its return is measured. When a target system makes online evaluation expensive (such as driving a robot car), unethical (such as treating a disease), or simply impractical (such as challenging a human chess master), effective offline evaluation metrics can play a critical role. In this paper, we compare several offline evaluation metrics, pointing out significant shortcomings that limit their utility. We propose a new metric we call "relative Bellman update error" (RBUE) that scores pairs of value functions using offline data. We provide analysis and empirical results that suggest the RBUE metric is a viable way of comparing value functions offline.

Keywords: Reinforcement Learning

1 Introduction

To assess the effectiveness of reinforcement-learning (RL) algorithms, it is important to have a way to compare pairs of algorithms head to head. The gold standard evaluation is to measure the return of each algorithm online (Kaelbling et al., 1996), declaring the algorithm that produced the highest return the winner.

While online evaluation is, in some sense, the only true measure of an algorithm's performance, there are many reasons to desire an offline evaluation metric using a fixed set of pre-collected data:

- If it is not possible to interact with the target system and no veridical simulator is available, offline evaluation is the only option.
- Even if a good simulator is available, if it is complex, time consuming, or expensive to use, offline evaluation might be preferred.
- Even if a simulator is available and easy to share, offline evaluation can be important for making the results more comparable. (Many research groups have rewritten incompatible versions of classic RL systems like "mountain car" due to language incompatibilities or other constraints.)
- When a target system is a human being (a patient with epilepsy or a grandmaster chess opponent, say) there are practical, and sometimes even ethical, concerns about testing learning results online. Offline evaluation can make it possible to collect the data once under controlled conditions and then share it with researchers throughout the community.

Evaluating learned classifiers offline using labeled batch data is standard practice in the supervised-learning community. The introduction of the UCI Machine Learning data repository (Newman et al., 1998) transformed the way machine-learning (ML) research is conducted and led to the development of multiple evaluation metrics as well as inspired other ML sub-communities to develop standards to collect data and evaluate performance (Bay et al., 2000).

Existing approaches for offline evaluation for RL have taken a number of forms. In the following section, we survey some of these approaches. We next present the properties of our proposed metric and then demonstrate its use to compare the performance of value functions learned in several benchmark systems.

2 Reinforcement-Learning Evaluation Metrics

We seek an evaluation metric that (1) is easy to calculate given batch data and (2) enables us to compare the performance of one state-action value function to another. That is, given a set of experience tuples of the form $\langle s, a, r, s' \rangle$ and a pair of state-action value functions Q_1 and Q_2, we want an evaluation of which value function is superior.

In this section, we survey some ways of assessing state-action value functions. The last metric is discussed in greater depth as we formulate our proposed metric. For consistency, we define each metric $M(Q_1, Q_2)$ so that it returns a positive value if Q_1 is believed to be superior, a negative value if Q_2 is believed to be superior, and zero if they are judged as being equally good. Given an experience tuple $\langle s, a, r, s' \rangle$, we define the *sample Bellman backup operator* $B^{s,a}_{r,s'}$ as a mapping from state-action value function Q to a new state-action value function that is identical to Q except

$$(B^{s,a}_{r,s'}Q)(s,a) = r + \gamma \max_{a'} Q(s', a'). \tag{1}$$

Using this notation, the standard Bellman backup operator B can be written as $(BQ)(s,a) = \{(B^{s,a}_{r,s'}Q)(s,a)\}^{s',r}_{T,R}$. The notation $\{\cdot\}^{s',r}_{T,R}$ denotes that we are taking an expected value of the expression $\cdot$ with $s' \sim T(s,a,s')$ and $r = R(s,a)$.

2.1 Short Survey of Studied Metrics

An ideal offline evaluation metric evaluates policies, but such metrics appear to require strong assumptions. We list a number of metrics and their properties:

- *Expected Return (Online)*: Even though it is an online metric, we use it as the gold standard. The most direct score for comparing state-action value functions is the expected return, return(Q), the discounted sum of rewards along an episode following the greedy policy $\pi_Q(s) = \text{argmax}_a\, Q(s,a)$. This policy is applied on the real system or simulator and the collected rewards are summed. Define $M_{\text{return}}(Q_1, Q_2) = \text{return}(Q_1) - \text{return}(Q_2)$.
- Fonteneau et al. (2010) developed *Model Free Monte Carlo-like policy evaluation* (MFMC). It is designed to evaluate a policy given a set of experience tuples, but we can use it to evaluate a value function Q by considering its greedy policy $\pi_Q(s) = \text{argmax}_a\, Q(s,a)$. The output, mfmc$(Q)$, is the return based on trajectories sampled using the policy and an approximate model of the transition dynamics. In particular, next states are chosen by finding

the closest one-step transitions in the set of experience tuples. Our approach differs from MFMC in that we develop an evaluation metric that uses the collected batch data directly, and not as a proxy to reconstruct the likely trajectories that would have been produced by the resultant policy. Define $M_{\text{MFMC}}(Q_1, Q_2) = \text{mfmc}(Q_1) - \text{mfmc}(Q_2)$.

- *Distance from Optimal Value Function*: The state-action value function Q^* is the solution to the equation $Q^* = BQ^*$. The greedy policy with respect to Q^* maximizes expected return. As such, it is natural to assess a state-action value function Q by its distance from Q^*. Using $||Q|| = \max_{s,a} |Q(s,a)|$ as the max norm of Q, we can evaluate Q via $||Q^* - Q||$. Singh and Yee (1994) relate the quantity $||Q^* - Q||$ to the difference in return between following Q's greedy policy and following Q^*'s. A positive property of this metric is that $M_{\text{dist}}(Q^*, Q)$ is non-negative for all Q. That is, no state-action value function is judged superior to Q^*. Its largest drawback is that it cannot be used unless Q^* is known, which will only be true for the most basic benchmark problems. Define $M_{\text{dist}}(Q_1, Q_2) = ||Q^* - Q_2|| - ||Q^* - Q_1||$.

- *Bellman Residual*: The Bellman backup of Q is $Q' = BQ$. As mentioned above, the optimal value function is obtained when $Q = Q'$, suggesting that the distance between these quantities, $||Q' - Q||$, sometimes called the *Bellman residual*, is another useful way of evaluating Q. Porteus (1982) provides an analysis that can be used to relate $||Q' - Q||$ to $||Q^* - Q||$ and therefore to the difference in expected return between following Q's greedy policy and following Q^*'s. A high value of the Bellman residual does not imply a poor value function. However, a value function with a lower Bellman residual has a tighter bound on its suboptimality. Its largest drawback is that it cannot be used unless the transition function is known (or is very densely sampled), because the transition function is needed for computing the Bellman backup. Define $M_{\text{residual}}(Q_1, Q_2) = \|BQ_2 - Q_2\| - \|BQ_1 - Q_1\|$.

2.2 Bellman Update Error

Consider again the Bellman residual and why it cannot be used without a model. For a value function Q, we would like to evaluate $||Q - BQ||$ where BQ is the Bellman backup applied to Q. That is, we want $Q(s,a)$ to be as close as possible to the average (over next states) value of $(B^{s,a}_{r,s'}Q)(s,a)$. By analogy to using samples to estimate an average, we define the following error measure we call *Bellman update error* (BUE) $\text{BUE}(Q) = \{\{(Q(s,a) - (B^{s,a}_{r,s'}Q)(s,a))^2\}^{s',r}_{T,R}\}^{s,a}_{\Pi}$. Here, state-action pairs are sampled from some probability distribution Π and next states are sampled from the transition function. The idea of Bellman update error is natural—it can be thought of as the Q-learning rule reconceptualized as an error measure—and has been used (explicitly or implicitly) repeatedly in the RL literature. We define the metric based on BUE as $M_{\text{BUE}}(Q_1, Q_2) = \text{BUE}(Q_2) - \text{BUE}(Q_1)$.

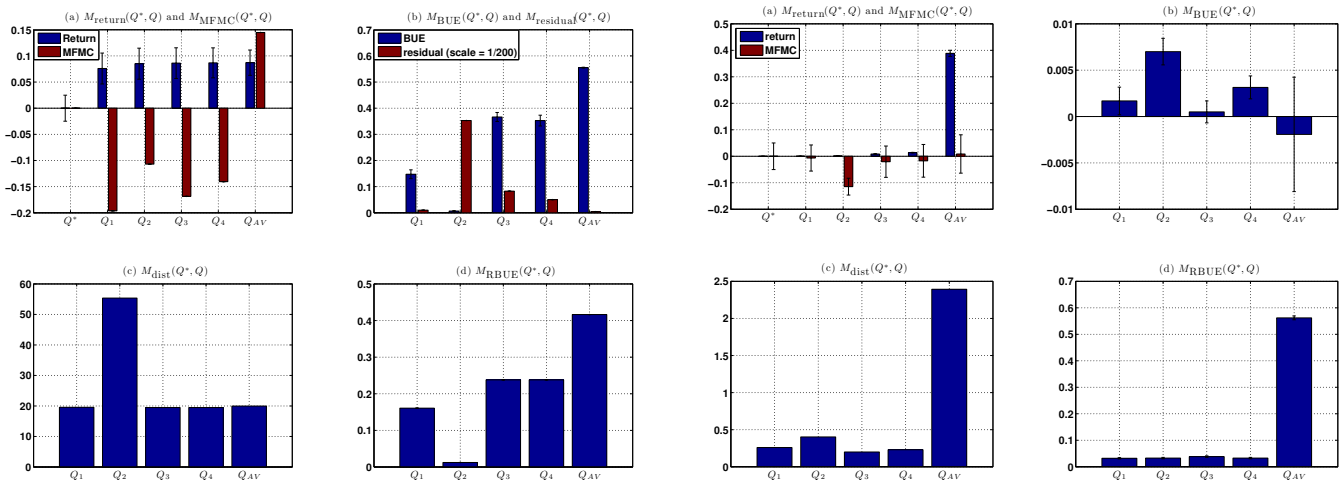
The BUE metric has the positive attribute that it can be estimated without knowing Q^* or T. Further, it can be estimated from sampled data as values are combined across state-action pairs sampled from a distribution Π. If test data is drawn from this distribution, approximating this average is straightforward. However, the Bellman update error is known to have some serious problems (Baird, 1995; Sutton and Barto, 1998). Its negative attribute is that, in stochastic systems, poor value functions can be rated as superior to the optimal value function because of an errant variance term in its formulation. In particular, given a distribution Π, we later show a stochastic system for which the value function $Q_{\text{AV}}(s,a) = \frac{\tilde{r}_\Pi}{1-\gamma}$ has the property that $\text{BUE}(Q_{\text{AV}}) < \text{BUE}(Q^*)$.

3 Relative Bellman Update Error

We propose the following novel metric for comparing two state-action value functions Q_1 and Q_2. Like BUE, it can be estimated from a sample of experience tuples. But, it has some important properties that set it apart from BUE. For example, it can produce a bound on the suboptimality of return, and it correctly selects Q^* as the optimal value function given the right sampling distribution Π.

The basic idea is to consider a Bellman update on the *average* of Q_1 and Q_2 instead of one or the other. For state-action value function Q_i (i is 1 or 2), we have $\text{RBUE}^{s,a}_i(Q_1, Q_2) = \{(Q_i(s,a) - (B^{s,a}_{r,s'}(Q_1 + Q_2))(s,a)/2)^2\}^{s',r}_{T,R}$. These quantities are then combined into a metric via $M_{\text{RBUE}}(Q_1, Q_2) = \{\text{RBUE}^{s,a}_2(Q_1, Q_2) - \text{RBUE}^{s,a}_1(Q_1, Q_2)\}^{s,a}_{\Pi}$.

The definition of M_{RBUE} removes the impact of the variance term that made M_{BUE} unreliable. A significant problem with the more direct metric M_{BUE} is that it can judge the optimal value function Q^* as being inferior to a very poor value function. We can show that a variant of M_{RBUE} that uses the max-norm instead of $\{\cdot\}^{s,a}_{\Pi}$ does not have this problem. The same result does not hold for $\{\cdot\}^{s,a}_{\Pi}$ with general distributions Π, but we conjecture that it holds when Π is the stationary distribution of the optimal policy.



(a) Metrics for Mountain Car

(b) Metrics for Marble Maze

Figure 1: Comparison of Evaluation Metrics for Mountain Car and Marble Maze Systems

4 Empirical Evaluation of Offline Evaluation Metrics

To assess the value of the different offline evaluation metrics, we carried out comparison experiments using a number of different reinforcement-learning systems. The metrics we used use for our evaluations were M_{return}, M_{dist}, M_{BUE}, M_{RBUE} and M_{MFMC}. The evaluations we report are on the systems Mountain Car (Sutton and Barto, 1998) and Marble Maze (Leffler et al., 2007).

4.1 Mountain Car

In Mountain Car, the goal is to get an underpowered car to reach the top of a hill. The system's state space has two dimensions (position and velocity) and three actions (forward throttle, backward throttle and no throttle). The system has a reward of 0.0 when the car reaches the top of the hill and a reward of -1.0 for every time point when the car is not at the top of the hill. In our configuration, the system is deterministic and the car always starts at the bottom of the hill.

The setup of the experiment was as follows. To calculate M_{BUE}, M_{RBUE} and M_{MFMC}, the batch data consisted of 5 sets of 30,000 transitions sampled from data collected from a suboptimal policy. The suboptimal, non-stationary policy took an average of 677 steps to reach the goal. For M_{MFMC} and M_{return}, we allowed a maximum of 1000 steps per episode. We evaluated 16 learned state-action value functions (8 from Q-learning, 8 from LSPI using Fourier basis), the optimal state-action value function Q^* and the constant function Q_{AV}, defined earlier. The metrics for a selected subset of state-action value functions are shown in Figure 1a. For this system, we also include M_{residual} to aid in better understanding of the results of the other metrics.

The optimal Q^* has the best performance in terms of return and the other value functions are all roughly equally sub-optimal. Interestingly, M_{MFMC} does a poor job of evaluating the value functions for this problem. Although it correctly identifies Q_{AV} as suboptimal, it estimates Q_1 through Q_4 as superior to Q^*. The metrics M_{dist}, M_{BUE} and M_{RBUE} are the "winners", as they all score Q^* as better than the other value functions. Note, however, that M_{BUE} and M_{RBUE} only barely prefer Q^* to Q_2. On the other hand, M_{dist} identifies Q_2 as quite a bit worse than the others, suggesting that the learner Q_2 fits the sampled test data well but that there are certain state-action values in the state-action space that it approximates poorly. Examining M_{residual} gives us insight into this issue. As M_{residual} covers the whole state-action space, it highlights which of the state-action value functions has the highest Bellman residual error over the whole state-action space. The M_{residual} metric prefers Q_2 least when compared to Q^*. This observation confirms that among the state-action pairs that were not in the sampled data, Q_2 has a poor approximation. The rest of the learners have relatively lower M_{residual}.

4.2 Marble Maze

The Marble Maze (Leffler et al., 2007) is a 2-dimensional discrete grid world with 81 states that includes a start state, pits, walls, and a goal. Its actions are *up*, *down*, *left* and *right*. Action effects are stochastic. A reward of -0.0001 is given for every timestep until the goal or a pit is reached. A reward of -1.0 was given for falling into a pit and a reward of 1.0 was given for reaching the goal.

The setup of the data collection and use for all of the metrics was similar to that of the Mountain Car system. The major difference was that a random policy with random starts states was used. The average collected trajectory length was 13.5 steps. The return was calculated from a single start state (Leffler et al., 2007). We create the same number and types of state-action value functions as described for Mountain Car. Comparisons for a subset of the state-action value functions is shown in Figure 1b.

In this system, all the value functions performed nearly optimally, with the exception of Q_{AV}, which was quite poor. The metric M_{MFMC} does not track M_{return} well, rating Q_{AV} as good compared to Q^* and Q_2 as better than Q^*. Here, M_{RBUE} and M_{dist} correctly assess the learned value functions as being near optimal and Q_{AV} as being poor. Again, as suggested by our analysis, the stochastic nature of this system causes trouble for M_{BUE}, which rates Q_{AV} as superior to Q^* due to its lower variance.

5 Summary and Conclusion

We conclude by summarizing our findings across all metrics. M_{return} is the gold standard online metric. Its major drawback is that it cannot be used offline—access to the real system or an accurate simulator is required. M_{dist} correlates reasonably well with M_{return} and can be applied offline. Its use, however, requires knowledge of Q^*, which can be extremely difficult to obtain. M_{MFMC} is an offline metric. It is easy to calculate and, in many cases, it can produces very accurate estimates of M_{return}. In general, it requires a good similarity function and sufficient testing data to adequately represent the state-action space. It also seems to have difficulties accurately simulating long trajectories, limiting its utility. M_{BUE} is an offline metric and is also very easy to compute. Its drawback is that it produces inaccurate evaluations in non-deterministic systems because of a sensitivity to variance. Our proposed metric, M_{RBUE}, is also easily computable offline. It is robust to non-deterministic settings and has the potential to be developed further. A shortcoming is that it can only make relative judgments between pairs of state-action value functions, making it impossible to produce a ranking of a collection of functions directly. Like all offline methods, it has a dependency on how testing data is collected—a biased sample can produce misleading evaluations.

The relative Bellman update error metric (M_{RBUE}) introduced in this paper could be used to build an online reinforcement-learning evaluation repository. We believe that, if sampled data sufficiently explores the state and action space, M_{RBUE} provides a foundation on which batch sampled data can be made available to researchers via the Internet and algorithms can be evaluated on common datasets without the complexity of creating and sharing simulators. The metric can be used completely offline by researchers to evaluate the algorithms they have or it can allow for a centralized service that could evaluate algorithms by comparing the state-action value functions they produce to other collected state-action value functions. Note that mechanisms for producing rankings from pairwise comparisons are well studied in the board game and sports communities; future work will examine adapting these schemes to the algorithm-evaluation setting.

References

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pp. 30–37.

Bay, S. D., D. Kibler, M. J. Pazzani, and P. Smyth (2000). The UCI KDD archive of large data sets for data mining research and experimentation. *ACM SIGKDD Explorations Newsletter - Special issue on "Scalable data mining algorithms"*, 81–85.

Fonteneau, R., S. Murphy, L. Wehenkel, and D. Ernst (2010). Model-free Monte Carlo-like policy evaluation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*.

Kaelbling, L., M. Littman, and A. Moore (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 237–285.

Leffler, B., M. Littman, and T. Edmunds (2007). Efficient reinforcement learning with relocatable action models. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 572.

Newman, D., S. Hettich, C. Blake, and C. Merz (1998). UCI Repository of Machine Learning Database, Irvine, CA: University of California, Dept. of Information and Computer Science.

Porteus, E. (1982). Conditions for characterizing the structure of optimal strategies in infinite-horizon dynamic programs. *Journal of Optimization Theory and Applications*, 419–432.

Singh, S. and R. Yee (1994). An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 227–233.

Sutton, R. and A. Barto (1998). *Reinforcement Learning: An Introduction*. Cambridge Univ Press.

Learned Myopic or Far-Sighted: Experience Shapes Human Temporal Horizon in Sequential Decisions

Hang Zhang
Psychology and Neural Science
New York University
hang.zhang@nyu.edu

Hyoseok Kim
Department of Psychology
New York University
hk1396@nyu.edu

Nathaniel D. Daw
Psychology and Neural Science
New York University
nathaniel.daw@nyu.edu

Laurence T. Maloney
Psychology and Neural Science
New York University
laurence.maloney@nyu.edu

Abstract

We investigated how well people make sequential decisions to achieve the long-term goal. In video-game-like settings, a spaceship flew across a row of three mountains of increasing heights. Before each mountain, subjects could elevate the spaceship by either a constant and small height (CS) or a variant but on average larger height (VL) to avoid crashing. The goal was to survive beyond the last mountain. The optimal choice before a specific mountain depended on the heights of all future mountains. We tested whether subjects could learn the optimal policy or base their choices only on a short horizon, i.e. on the immediate mountain.

Methods: We constructed two combinations of mountain heights, A & B, which differed in how early a short horizon would be penalized. For A, a short horizon would yield the optimal choice before the first mountain and not increase crash rate until the last mountain. In contrast, for B, a short horizon would increase crash rate as early as the second mountain. Each subject completed 4 blocks of 60 trials, in the block order of ABAB or BABA. Sixteen naïve subjects were evenly assigned to the two groups.

Results: The two groups differed in their learning trajectories. (1) The ABAB group achieved a higher probability of survival in the last (.63) than in the first two blocks (.50), but the BABA did not (both .54). (2) The ABAB had a shorter horizon than the BABA: When VL was the optimal choice and involved long-term considerations, ABAB chose VL less than BABA did (53% vs. 67%). (3) The BABA appeared to be far-sighted: When CS was the optimal choice and reduced crash at the immediate mountain, BABA chose CS less than ABAB did (60% vs. 81%).

Conclusion: Human individuals' temporal horizon in a sequential-decision task depends on their initial experience with the task. People may learn to be myopic or far-sighted.

Keywords: sequential decisions; temporal horizon; decision under risk; order effect

Acknowledgements

The project was partly supported by NIH EY019889.

We investigated how well people make sequential decisions to achieve the long-term goal. In video-game-like settings (Fig. 1), a spaceship moved from left to right to approach a row of three mountains of increasing heights. It would crash into a mountain if it was still lower than the mountain upon arrival. Before each mountain, subjects could elevate the spaceship by either a constant and small height (CS) or a variant but on average larger height (VL). The goal of each trial was to survive beyond the last mountain. Subjects received monetary rewards for each survival. The optimal choice before a specific mountain depended on the heights of all future mountains. We tested whether subjects could learn the optimal policy or base their choices only on a short horizon, i.e. on the immediate mountain.

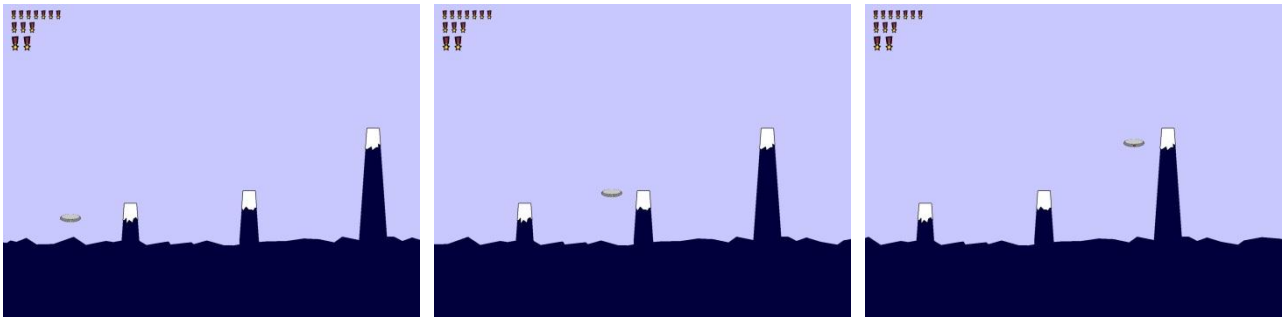


Figure 1. **Screenshots of a trial**.

Methods: Before each mountain, subjects had up to 2.5 seconds to choose CS or VL by key press. CS was 2 units. VL was equally likely to be 1 or 5 units. The experiment consisted of three stages. Subjects first sampled each option for 24 times and observe their outcomes. Then they chose between CS and VL to survive one mountain of 1.5, 2.5, 3.5, or 4.5 units high, for 80 trials.

In the following main task, subjects aimed to survive three mountains, whose heights were [1.5, 2.5, 7.5] (Combination A) or [0.5, 5.5, 7.5] (Combination B). Combinations A & B differed in how early a short horizon would be penalized. For A, a short horizon would yield the optimal choice before the first mountain and not increase crash rate until the last mountain. In contrast, for B, a short horizon would increase crash rate as early as the second mountain. Each subject completed 4 blocks of 60 trials, in the block order of ABAB or BABA. Sixteen naïve subjects were evenly assigned to the two groups.

Results: The two groups differed in their learning trajectories. (1) The ABAB group achieved a higher probability of survival in the last (.63) than in the first two blocks (.50), $F(1,21)=13.05$, $p=.002$, but the BABA did not (both .54). See Fig. 2a. (2) The ABAB had a shorter horizon than the BABA: When VL was the optimal choice and involved long-term considerations, ABAB chose VL less than BABA did (51% vs. 60% before B's first mountain, 55% vs. 73% before A's second mountain, $F(1,57)=2.99$, $p=.089$). See Fig. 2b, middle and right. (3) The BABA appeared to be far-sighted: When CS was the optimal choice and reduced crash at the immediate mountain, BABA chose CS less than ABAB did (60% vs. 81% before A's first mountain, $F(1,28)=8.51$, $p=.007$). See Fig. 2b, left. We verified that the two groups did not differ in their performances in the one-mountain task.

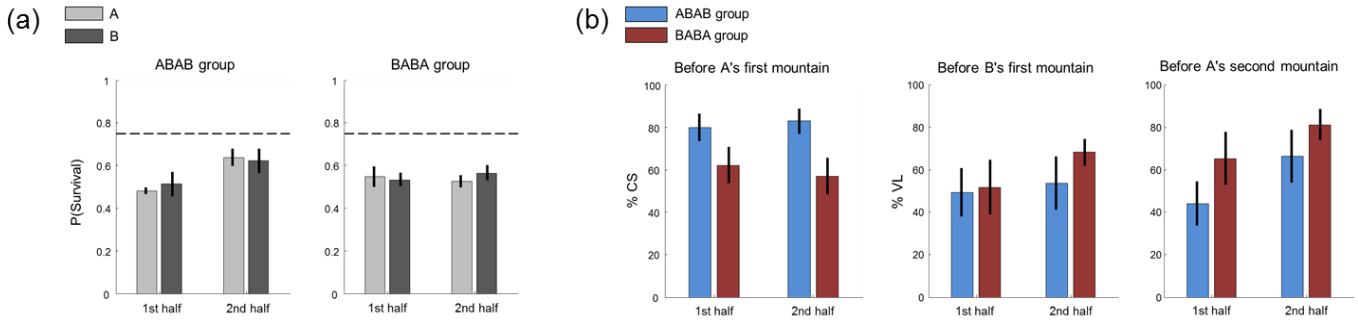


Figure 2. **ABAB group vs. BABA group in the three-mountain task**. (a) Probability of survival for the two groups and for the 1st and 2nd half of trials. Dash line denotes the maximum expected probability of survival. (b) Percentage of trials of choosing CS when CS was the optimal choice and reduced immediate crash (left panel) and of choosing VL when VL was the optimal choice and facilitated surpassing future mountains (middle and right panels). Error bars denote 1 SE.

Conclusion: Human individuals' temporal horizon in a sequential-decision task depends on their initial experience with the task. People may learn to be myopic or far-sighted.

Manipulating model-based and model-free control through neurostimulation of prefrontal cortex

Peter Smittenaar
WTCN
UCL
petersmittenaar@gmail.com

Thomas H.B. FitzGerald
WTCN
UCL
thbfitz@gmail.com

George Prichard
ICN
UCL
d.g.m.prichard@gmail.com

Vincenzo Romei
WTCN
UCL
v.romei@ucl.ac.uk

Nicholas D. Wright
WTCN
UCL
n.wright@ucl.ac.uk

Joern Diedrichsen
ICN
UCL
j.diedrichsen@ucl.ac.uk

Raymond J. Dolan
WTCN
UCL
r.dolan@ucl.ac.uk

WTCN: Wellcome Trust Centre for Neuroimaging. UCL: University College London, UK

Abstract

Human choice behavior often reflects a competition between inflexible but computationally efficient control on the one hand and slower but more flexible systems of control on the other. This distinction is well captured by model-free and model-based reinforcement learning algorithms, which share many similarities with habitual and goal-directed behaviors, respectively. These two systems often compete for control over choice, and it has been suggested that an imbalance between controllers might underlie a wide range of disorders, including addiction and Parkinson's disease. Causally manipulating this balance in humans will provide insight into the neural structures underlying value-based choice, and serve as a potential avenue for intervention in disorders of these systems. Here we studied human subjects performing a task that allows the quantification of model-based and model-free control (Daw et al., 2011, Neuron), following theta-burst transcranial magnetic stimulation (TMS) to the right or left dorsolateral prefrontal cortex, or the vertex. We show it is possible to shift the balance of control between these systems by disruption of dorsolateral prefrontal cortex, such that participants manifest a dominance of simpler but less optimal model-free control, compared to vertex. We will also present data on the same task from an enhancement, rather than impairment, of dorsolateral prefrontal cortex processing through transcranial direct current stimulation.

Keywords: model-based control, prefrontal cortex, transcranial magnetic stimulation

Acknowledgements

We acknowledge support from the Wellcome Trust (RJD, Wellcome Trust Senior Investigator award 098362/ Z/ 12/ Z; PS, 4-year PhD studentship; The Wellcome Trust Centre for Neuroimaging is supported by a Wellcome Trust Strategic Grant 091593/ Z/ 10/ Z).

1 Introduction

An elegant computational framework that captures the presence of (often competing) habit-like and goal-directed behaviors is provided by model-free and model-based control[1,2]. A model-free system learns a single value for each action based on reward prediction errors and guides behavior based on these alone, requiring a minimum of computational effort at a cost of a lack of flexibility in adjusting to current goals. Model-based control, by contrast, dynamically computes optimal actions, a process that is computationally demanding but allows for flexible, outcome-specific behavioral repertoires. In this study we focused on the involvement of the dlPFC in model-based control. We focused on this region based on previous evidence for its role in the construction and use of associative models[3-5] and the coding of hypothetical outcomes[6]. Work on non-human primates also implicates the dlPFC as a site for convergence of reward and contextual information[7]. However, the key human evidence for dlPFC involvement in model-based control has been based on correlational evidence using functional imaging (fMRI) or single-unit recordings. Here we describe two studies in which we attempted to shift the balance between these two systems in human participants by neurostimulation of the dlPFC. We predicted that a disruption of dlPFC through theta burst transcranial magnetic stimulation (TBS[8]) should selectively impair model-based control and thus lead to a relative shift towards model-free control; conversely, an enhancement of dlPFC through anodal transcranial direct current stimulation (tDCS[9]) should lead to an enhancement of model-based relative to model-free control.

2 Methods

2.1 Participants and stimulation protocol

For the TBS study, we recruited 25 human participants (mean age (SD): 24.2 (4.0) years). These participants were tested on 3 separate sessions (3 to 16 days apart) after off-line MRI-guided TBS to the right dlPFC, left dlPFC, or vertex (control site; intersect between nasion-inion line running front-to-back over the skull, and line running between ears over the top of the head). The TBS protocol leads to reduced excitability of the underlying neural tissue for a period of at least 20 minutes[8], which is thought to interfere with normal function. For the tDCS study we recruited a different group of 23 human participants (mean age (SD): 22.5 (5.3) years). All participants were tested in a double-blind design on 2 sessions 3-6 days apart with on-line sham or active anodal stimulation to right dlPFC, and cathodal stimulation over the inion. A direct current of 2 mA leads to increases in excitability of the tissue underneath the anode[9]. One participant was excluded from the tDCS study because stimulation was interrupted during the task due to a decrease in conductance over time.

2.2 Task

We used a task that enables quantification of model-based and model-free control over choices[10]. Participants were required to make two choices on every trial to arrive at a rewarded or a non-rewarded outcome (Figure 1A). Choices at the first stage of the task probabilistically determine which pair of options becomes available at the second stage. For each first stage action, one pair of second-stage options is more likely to occur (a 'common transition'). Because a model-based controller is able to incorporate the probability of state-state transitions into its decision making, whilst the model-free controller is not, the predictions made by these controllers diverge after uncommon transitions (Figure 1B). For example, a reward obtained after an uncommon transition prompts a model-free agent to (erroneously) choose the same first-stage stimulus on the next trial, since action values are updated based solely on the rewards that follow the action. In contrast, a model-based agent would be more likely to switch to the previously unchosen first-stage stimulus, as such a switch will make it more likely that the agent arrives at the rewarded second state again. Using these divergent predictions about first-stage choice behavior, we can quantify the influence of the controllers in terms of the main effect of reward (model-free) and the interaction between reward and transition likelihood (model-based) on the probability of staying with the same first-stage stimulus (Figure 1B). Following instruction and 50 practice trials using a different set of stimuli, participants completed 201 trials for the TBS study, or 350 trials in the tDCS study.

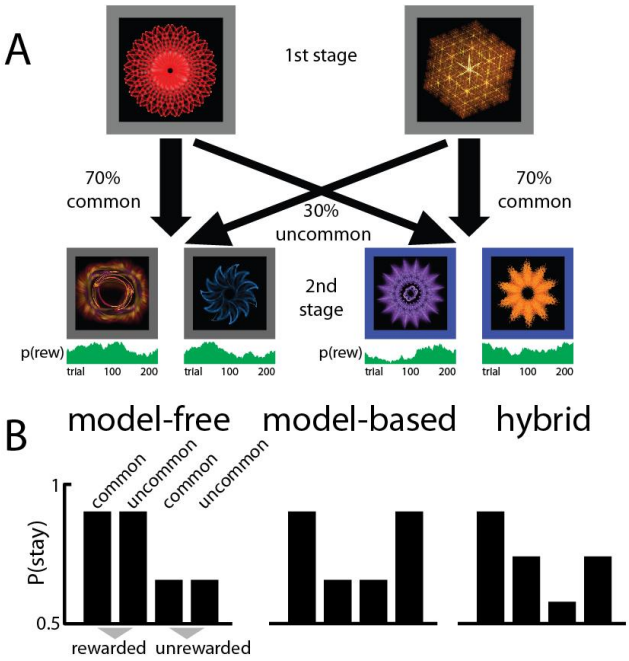


Figure 1: Task design (a) On each trial a choice between two stimuli led probabilistically to one of two further pairs of stimuli, which then demanded another choice followed by reward or no-reward according to the p(reward) of the chosen second-stage stimulus that fluctuated over time. (b) Model-based and model-free strategies for reinforcement learning predict differences in feedback processing particularly after uncommon transitions. We can thus quantify model-free control by estimating the main effect of reward, and model-based control by estimating the reward-by-transition interaction.

2.3 Analysis

We estimated the main effect of reward (model-free control) and the reward-by-transition interaction (model-based control) for each experimental condition using hierarchical logistic regression, with all coefficients taken as random effects across participants. In the TBS study this regression only included events on the previous trial, whereas in the tDCS study we added regressors examining effects of reward and reward-by-transition up to 5 trials in the past. Planned contrasts were performed on these regression coefficients to examine within-subject effects of stimulation for the TBS and tDCS studies separately. The analyses were performed in Matlab and using the LME4 toolbox for R.

3 Results

3.1 TBS to right dlPFC disrupts model-based control

We observed positive coefficients for the reward and reward-by-transition regressors for all three TBS sites (all $p < .006$), confirming that behavior comprised a hybrid of model-free and model-based control. Levels of model-based and model-free control after left and right dlPFC TBS were then contrasted with vertex (Figure 2A). We observed that TBS to neither left ($p = .52$) nor right ($p = .20$) dlPFC significantly changed model-free control compared to vertex. By contrast, model-based control was disrupted following TBS to right ($p = .01$) but not left ($p = .89$) dlPFC compared to vertex. We observed no difference in model-based control between left and right dlPFC ($p = .13$). We also computed a measure of the relative balance between these two systems as $\beta_{\text{model-based}} - \beta_{\text{model-free}}$ (Figure 2B). This showed a significant shift towards model-free control caused by TBS to right ($p = .01$) but not left ($p = .63$) dlPFC compared to vertex. We observed no difference between left and right dlPFC ($p = .11$).

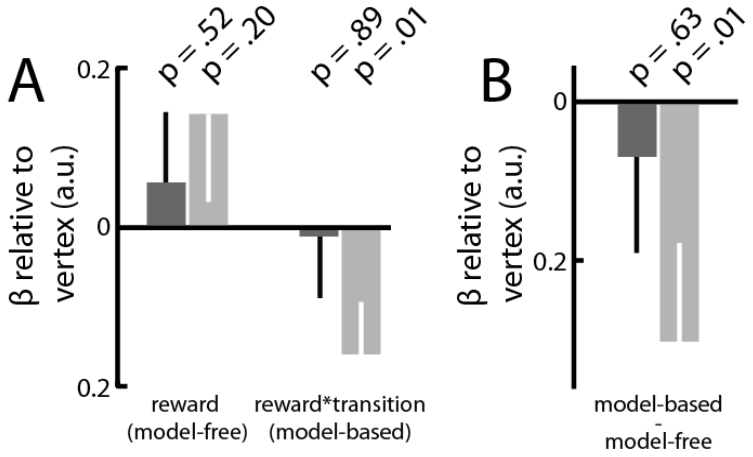


Figure 2: Results (a) Disruption of right dlPFC reduced model-based control compared to vertex. TBS did not significantly affect model-free control. (b) The balance between the controllers was calculated as $\beta_{model\text{-}based} - \beta_{model\text{-}free}$. The balance significantly shifted towards model-free control after disruption of right, but not left, dlPFC compared to vertex. Error bars indicate SEM.

3.2 Effect of TBS to left dlPFC interacts with working memory capacity

Model-based control is thought to depend on prefrontal working memory (WM). Given that studies of WM observe lateralized functionality (e.g.[11]) we asked whether the magnitude of the TBS effect might be related to WM capacity. To examine such inter-individual differences we could not use the population parameter estimates obtained through the regression. Instead, we extracted the numerical magnitude of the main effect of reward, the reward-by-transition interaction and the difference between the two from each subject's average stay probability in each of the four reward/ transition conditions in each stimulation condition. We correlated the balance between the two systems in all stimulation conditions with WM. Strikingly, only behavior after disruption of left dlPFC was WM-dependent (Figure 3; vertex, r = .09, p = .68; left dlPFC r = .53, p = .006; right dlPFC, r = -.05, p = .80). Pairwise permutation tests revealed the correlation was significantly more positive in left compared to right dlPFC (10^5 permutations, p = .009), marginally more positive in left dlPFC compared to vertex (p = .06), and not significantly different between right dlPFC and vertex (p = .52). Taken together, these data show that the effect of left dlPFC disruption on the balance between model-based and model-free control depends on WM capacity, with high WM participants retaining more model-based control compared to those with low WM.

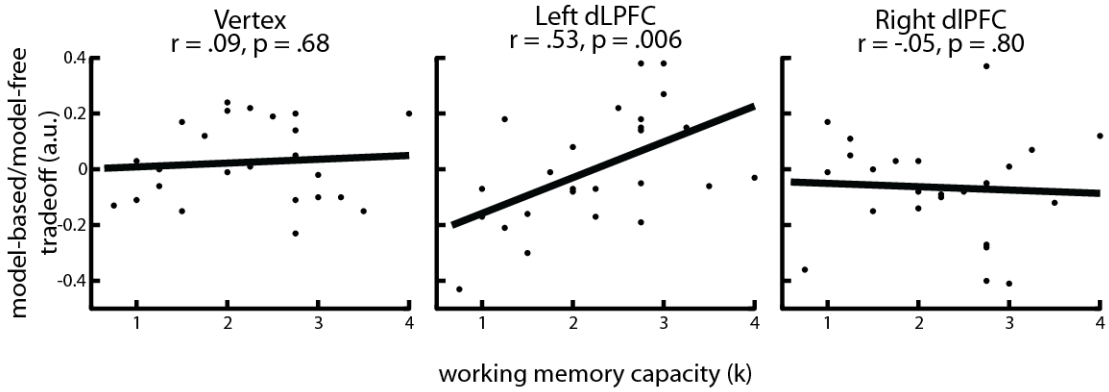


Figure 3: Working memory (WM) capacity did not predict the balance between model-based and model-free control after disruption of vertex (left) or right dlPFC (right). In contrast, higher WM was associated with relatively stronger model-based control after disruption of left dlPFC (middle).

3.1 Preliminary analysis of anodal tDCS to right dlPFC

The tDCS data are in a preliminary stage of analysis. We performed a regression as in the TBS study, but now attempting to predict the current choice based on events up to 5 trials in the past. Firstly, we observe that significant model-free and model-based influences on choice occur as far back as 5 trials in the past, in both sham and active stimulation (each coefficient > 0, each p < .05; Figure 4). Where we predicted the

Active condition to reveal stronger model-based influences on behavior, this was not significant when taken over the 5 trials together (p = .80), on any of the individual time points (all p > .34), or when examined as an interaction with Lag (p = .85). Rather, it seemed that model-free control was somewhat stronger in the Sham compared to Active condition, though the statistical evidence in this preliminary analysis is weak (Lag-1 model-free coefficient Sham > Active, p = .07; Lag-3 model-free coefficient Sham > Active, p = .10)

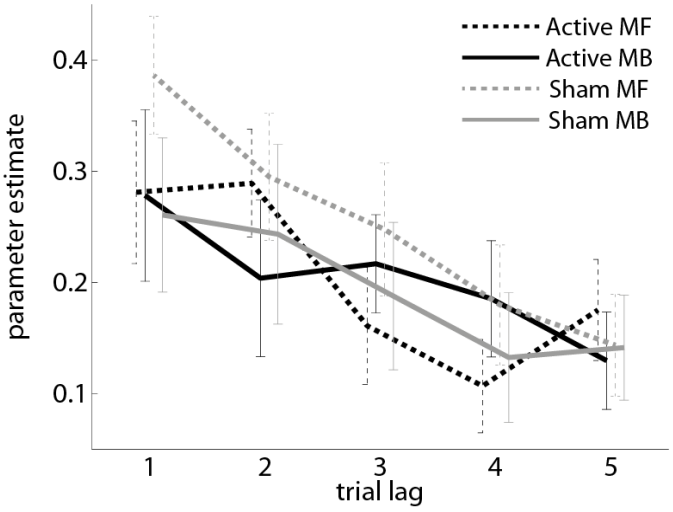


Figure 4: Behavior on trial n explained by events on trial n-1 to n-5. All 5 Lag trials significantly contribute to choice behavior, suggesting extended integration of information in both the model-free and model-based system. No significant effects of stimulation condition (Active versus Sham) were found. MF = model-free, MB = model-based, error bars indicate ± SEM.

4 Discussion

These data show that a disruption of the right dlPFC leads to a selective impairment of model-based control, suggesting a necessary role for this region in complex, goal-directed action. Disruption of the left dlPFC only impaired model-based control in individuals with low working memory capacity, suggesting a protective effect of high working memory on transient disruption of left dlPFC function. It is not yet clear why an enhancement of right dlPFC using tDCS does not elicit improved model-based control, though further analyses on these data are needed to explore this manipulation fully.

5 References

1 Daw, N. D., Niv, Y. & Dayan, P. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat Neurosci* **8**, 1704-1711 (2005).
2 Dayan, P. & Niv, Y. Reinforcement learning: the good, the bad and the ugly. *Current Opinion in Neurobiology* **18**, 185-196 (2008).
3 Wunderlich, K., Dayan, P. & Dolan, R. J. Mapping value based planning and extensively trained choice in the human brain. *Nat Neurosci* **15**, 786-791, doi:10.1038/nn.3068 (2012).
4 Gläscher, J., Daw, N., Dayan, P. & O'Doherty, J. P. States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning. *Neuron* **66**, 585-595 (2010).
5 Xue, G., Juan, C. H., Chang, C. F., Lu, Z. L. & Dong, Q. Lateral prefrontal cortex contributes to maladaptive decisions. *Proc Natl Acad Sci U S A* **109**, 4401-4406, doi:10.1073/pnas.1111927109 (2012).
6 Abe, H. & Lee, D. Distributed coding of actual and hypothetical outcomes in the orbital and dorsolateral prefrontal cortex. *Neuron* **70**, 731-741 (2011).
7 Lee, D. & Seo, H. Mechanisms of reinforcement learning and decision making in the primate dorsolateral prefrontal cortex. *Ann N Y Acad Sci* **1104**, 108-122, doi:10.1196/annals.1390.007 (2007).
8 Huang, Y. Z., Edwards, M. J., Rounis, E., Bhatia, K. P. & Rothwell, J. C. Theta burst stimulation of the human motor cortex. *Neuron* **45**, 201-206, doi:10.1016/j.neuron.2004.12.033 (2005).
9 Nitsche, M. A. *et al.* Transcranial direct current stimulation: State of the art 2008. *Brain stimulation* **1**, 206-223 (2008).
10 Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P. & Dolan, R. J. Model-based influences on humans' choices and striatal prediction errors. *Neuron* **69**, 1204-1215 (2011).
11 Mull, B. R. & Seyal, M. Transcranial magnetic stimulation of left prefrontal cortex impairs working memory. *Clinical Neurophysiology* **112**, 1672-1675 (2001).

Reinforcement learning and novelty seeking across the lifespan

Audrey Houillon
(1,2)
audrey.houillon@tu-berlin.de

Robert Lorenz
(3,4)
robert.lorenz@charite.de

Tobias Gleich
(3)
tobias.gleich@charite.de

Jürgen Gallinat
(3)
juergen.gallinat@charite.de

Andreas Heinz
(3)
andreas.heinz@charite.de

Klaus Obermayer
(1,2)
oby@ni.tu-berlin.de

1. Bernstein Center for Computational Neuroscience Berlin, Germany,
2. Neural Information Processing Group, Department of Software Engineering and Theoretical Computer Science, Technical University Berlin, Berlin, Germany,
3. Clinic for Psychiatry and Psychotherapy, Charite University Medicine,Berlin, Germany,
4. Department of Psychology, Humboldt-Universitaet zu Berlin, Germany

Abstract

We investigated how reward learning and its interaction with novelty-seeking could be affected across the lifespan. Stimulus novelty enhances exploratory choices through engagment of neural reward systems. As these reward systems depend on dopamine, which in turn has been proposed to decrease with increasing age, we hypothesized that aging may be associated with changes in reward learning processes. We applied a reward-dependent learning task to younger and older groups. Computational models were used to quantify differences in behavioral performance and brain activation (fMRI). We showed that novel stimuli presented from a pre-familiarized category could accelerate or decelerate learning of the most rewarding category, depending on whether novel stimuli were presented in the best or worst rewarding category. The extent of this influence depended on the individual trait of novelty seeking. For novelty seekers, learning was accelerated in the best category and decelerated in the worst category, when novelty was presented. The opposite effect could be observed for novelty avoiders. Subjects' choices were quantified in reinforcement learning models, including a parameter to characterize individual variation in novelty response. The theoretical framework further allowed us to test different assumptions, concerning the motivational value of novelty. fMRI analysis showed the strongest signal change in the condition where reward and novelty were presented together, but only in low probability of correct action trials. This effect was observed in the striatum, midbrain and cingulate cortex. The model also showed that older subjects had lower novelty seeking behavior, but overall explorative behavior was increased.

Keywords: hidden markov model; novelty seeking; aging

Acknowledgements: This study was supported by the German Ministry for Education and Research(BMBF 01GQ0914/01GQ1001B/01GQ0911) and a German National Academic Foundation grant to R.C. Lorenz.

1 Extended Abstract

Human adaptive behavior changes across the lifespan and is strongly linked to the functioning of the dopaminergic system [2-4]. Here we investigate how the hypothetical alteration of dopaminergic functioning could affect novelty-seeking behavior and its interaction with reward-learning across the lifespan. Dopamine synthesis in striatum and midbrain has a tendency to decrease with increasing age [18]. Novel stimuli tend to be associated with a stronger explorative behavior in the context of a reward-based learning paradigm in humans [7]. Stimulus novelty enhances these exploratory choices through engagment of neural reward systems, that have been shown to depend on dopamine. Thus we hypothesize that modifications of the dopaminergic system across the lifespan may be associated with changes in learning processes, changes that have already been observed both on behavioral [19] and neurobiological [20] levels. We apply a reward-related learning tasks with novelty interaction to younger and older subject groups, which are expected to have different learning profiles a consequence of aging. Data interpretation crucially depends on computational models. These are used to quantify differences in behavioral performance and brain activation (fMRI).

1.1 Novelty-seeking across the lifespan

Recent research suggests that novelty has an influence on reward-related decision-making [7-13]. Novel stimuli tend to be associated with a stronger explorative behavior in the context of a reward-based learning paradigm in humans [7]. Stimulus novelty enhances these exploratory choices through engagment of neural reward systems, such as the ventral striatum (VS) and the ventral tegmental area (VTA) which are mesolimbic dopaminergic structures [7-11]. Thus we hypothesized that reward learning would be affected by novelty and that this interaction would be modified with increasing age.

In this study, 22 healthy young human subjects (mean age 25) were presented with a probabilistic reinforcement learning task (see Figure D). Novel stimuli were presented in 30% of the trials, within the high reward category or the low reward category, depending on the condition. Conditions were clearly delimited. Here, we showed that novel stimuli presented from a pre-familiarized category could accelerate or decelerate learning of the most rewarding category, depending on whether novel stimuli were presented in the best or worst rewarding category (see Figure B). The extent of this influence depended on the individual psychological trait of "novelty seeking" (see Figure C). Subjects' choices were quantified in reinforcement learning models. We introduced a bias parameter to model exploration toward novel stimuli and characterize individual variation in novelty response. The theoretical framework further allowed us to test different assumptions, concerning the motivational value of novelty [1]. One the one hand SN/VTA activation by novelty in a rewarding context has raised the possibility that novelty per se might have intrinsic rewarding properties. On the other hand specific SN/VTA activations to novelty alone suggest a second, reward-independent mechanism, that favors exploration toward the novel cue [9-11]. Based on these findings, we proposed that novelty per se can act as an explorative bias, but can also act as a bonus for rewards when these are explicitly attended [12] (see Figure A). Individual variation in novelty response were characterized by finding the best-fitting values of parameters for each subject. The best-fitting model was a hidden markov model that combined both novelty components. Here again, the model's bias parameter also showed a significant correlation with the independent novelty-seeking trait.

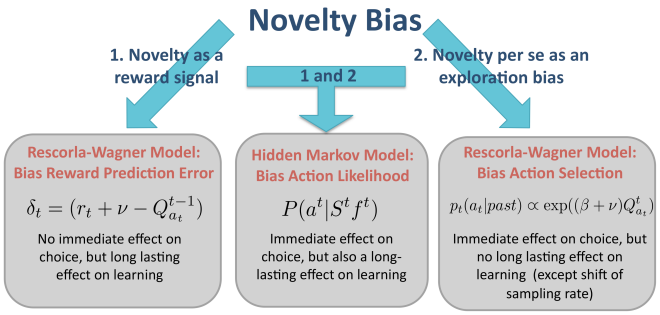
Additional fRMI analysis was performed in a model-based fashion, by including the time series for the probability of correct action predicted by the bestfitting reinforcement learning model. A three-way ANOVA with factors condition, novelty and action probability was conducted. The computed probability of correct action was used to identify trials with a high probability of correct action and a low probability of correct action. Results showed the strongest signal change in the condition where reward and novelty were presented together, but only in low probability of correct action trials. These results are in line with previous research [13], which showed that striatal reward processing could be boosted by novelty, but only in uncertain states during explorative behavior and acquisition phase. This effect was furthermore observed in the midbrain (origin of dopamine), cingulate cortex and PFC (both projection areas of dopamine pathways).

It is to be noted that the cingulate cortex is a major projection site of the locus corelueus, the main site of the noradrenaline system. Noradrenaline is believed to be associated with explorative behaviors [14-17].

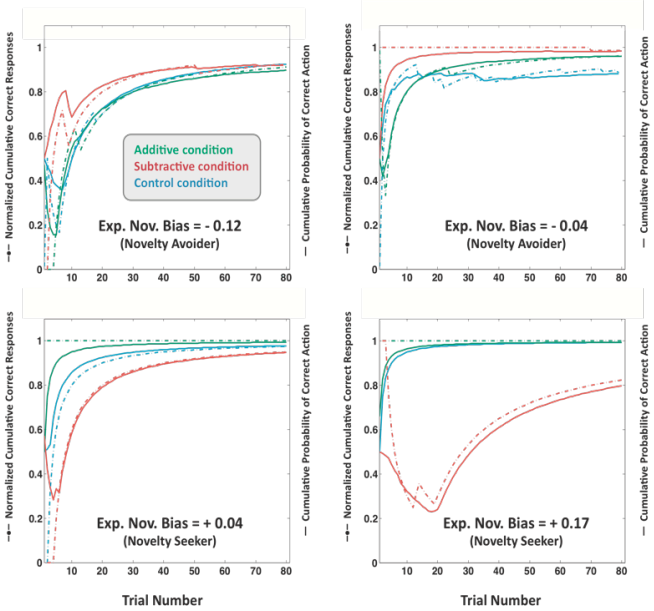
The magnitude of the personality trait "novelty-seeking" decreases across the lifespan [21-22]. We therefore expect that the novelty-seeking bias in particular should be lowered in the older group, however general explorative behavior should increase as a consequence of age-induced diminution of dopamine behaviors [17]. In order to test these assumptions, a slightly modified version of the novelty paradigm was conducted with 25 healthy young subjects (Mean Age=26) and 24 healthy older subjects (Mean Age=66), with the additional differences being that now single trials from different conditions were presented in a randomized order. Older subjects tend to have lowered levels of dopamine and glutamate and lower novelty-seeking scores, leading to our hypothesis that novelty seeking behavior should be lowered, but overall explorative behavior increased [cf. 17]. The model parameters of the bestfitting model (here again the hidden markov model with both novelty effects) behaved according to hypothesis: the computational parameters of novelty seeking were significantly lower, whereas the exploration parameters were significantly higher in the older as compared to the younger subject group. The fMRI analysis remains to be conducted, in order to test whether neural activations are in accordance with the behavioral results.

1.4 Figures

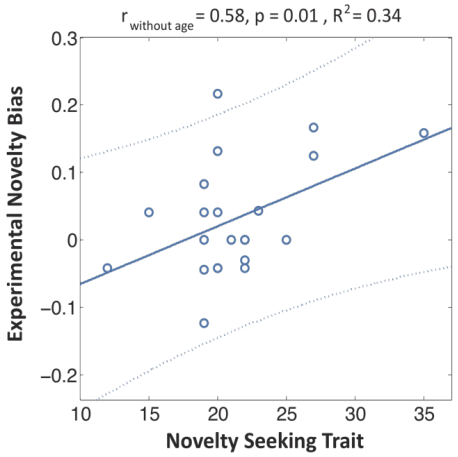
(A)



(B)



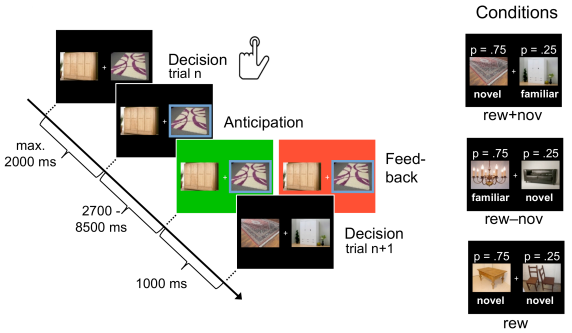
(C)



((A) Hypothetical effects of novelty and their computational implementation in different reinforcement learning models (B) Correlation between personality trait "novelty seeking" and experimental novelty bias from the paradigm ($CR_{"reward+novelty"\ condition} - CR_{"reward-novelty"\ condition}$, from novelty trials only). In addition, we found a significant correlation between the novelty bias parameters from the best-fitting model and both the experimental novelty bias and novelty seeking trait (not shown here). (C) Experimental and simulated learning curves for individual subjects. (Dotted lines: normalized cumulative correct responses (CR) over trial time. Solid line: simulated normalized cumulative probability of choosing the correct category. For novelty seekers (positive novelty

bias), the correct category seems to be learned faster in the "reward+novelty" (additive), and slower in the "reward-novelty"(subtractive), - at least in the initial learning phase- as compared to the control condition. The opposite is true for novelty avoiders (negative bias). (D) Experimental paradigm

(D)



1.4 References

[1] Houillon A, Lorenz RC, Boehmer W, Rapp MA, Heinz A, Gallinat J, Obermayer K (2013) Progress in Brain Research. 202: 415-440.
[2] Adcock RA, Thangavel A, Whitfield-Gabrieli S, Knutson B, Gabrieli JD. (2006). Neuron 50, 507-17.
[3] Dreher JC, Meyer-Lindenberg A, Kohn P, Berman KF. (2008) Proc Natl Acad Sci USA, 105, 15106-11.
[4] Frank MJ, Kong L.Psychol Aging. 2008 Jun;23(2):392-8.
[5] Hazy T, Frank M, O'Reilly R. (2007) Phil. Trans. Roy. Soc. B 2007
[7] Wittmann BC, Daw ND, Seymour B, Dolan RJ (2008) Neuron, 58(6), 967-973.
[9] Krebs RM, Schott BH, Düzel E (2009 a) Biological Psychiatry, 65(2), 103-110.
[10] Ruth M Krebs, Björn H Schott, Hartmut Schütze, Emrah Düzel (2009 b) Neuropsychologia, 47(11),2272-2281.
[11] Krebs RM, Heipertz D, Schuetze H, Duzel E. (2011) Neuroimage, 58(2),647-655.
[12] Kakade S, Dayan P (2002) Neural networks, 15(4-6), 549-559.
[13] Guitart-Masip et al (2010) J Neuroscience 30(5), 1721-1726
[14] Nassar MR, Rumsey KM, Wilson RC, Parikh K, Heasly B, Gold JI.(2012) Nat Neurosci. 2012 Jun 3;15(7)
[15] Cohen JD, Mc Clure SM, Yu A (2007) Phil. Trans. R. Soc. B (2007) 362, 933–942
[16] Jepma M, Nieuwenhuis S. (2011) J Cogn Neurosci. 2011 Jul;23(7):1587-96
[17] Doya K (2008) Nat Neurosci. Nat Neurosci. Apr;11(4):410-6
[18] Kumakura, Y., Vernaleken, I., Buchholz, H.-G., Borghammer, P., Danielsen, E., Grunder, G., Heinz, A (2010).Neurobiology of aging, 31(3), 447–463.
[19] Frank, M. J., & Kong, L. (2008).Psychology and aging, 23(2), 392–398.
[20] Pietschmann, M., Endrass, T., Czerwon, B., & Kathmann, N. (2011). Biological psychology, 86(1), 74–82.
[21] Otter C, Huber J, BonnerA (1995) 18 (4), 471–480.

[22] Weyers P., Krebs H, Janke W (1995) Pers Individual Differ. 19 (6), 853–861.

Social Reinforcement For Collective Decision-Making Over Time

Marco A. Montes de Oca*
Department of Mathematical Sciences
University of Delaware
Newark, DE 19716
mmontes@math.udel.edu

Abstract

Social interactions underpin collective decision making in all animal societies. However, the actual mechanisms that animals use to achieve a collective decision differ among species. For example, ants use pheromones to bias the decisions of other ants; birds observe and match the velocity of their neighbors; humans match the speed of other people while driving (even above speed limits). Despite the differences among these (and other) collective decision-making mechanisms, some basic principles underlying them exist, like the tendency to conform to the actions or opinions of others. In our examples, by following pheromone trails ants effectively follow on their nestmates' steps, birds in a flock are more likely to fly in the same direction, and we humans, while we do not always agree, we do not like to be in permanent conflict with others and eventually seek ways to compromise. Therefore, this principle's basic operating mechanism is that an individual who is exposed to the actions or opinions of others tends to perform the actions, or have the same opinions of the observed individuals.

In this communication, I describe two basic collective decision-making mechanisms based on the principle outlined above. These mechanisms are tested in a setting that simulates a robotics scenario in which a group of robots must collectively find the shorter of two alternative paths between two areas without measuring travel times or distances. First, I describe a mechanism that consists of robots forming teams of three robots (or a greater odd-number of robots) that decide which path to use by locally using the majority rule. Then, I describe a mechanism that consists in individual robots increasing the tendency to choose either path based on the path recently taken by another robot. In both cases, the group collectively chooses the shorter path with high probability. These mechanisms have been proposed as swarm intelligence mechanisms for optimal collective decision-making.

Keywords: Social Reinforcement, Collective Decision Making, Swarm Intelligence

*Homepage: http://www.math.udel.edu/~mmontes/

1 Introduction

There are many animal species whose members gather in large numbers and make important decisions as a group, that is, they make collective decisions [1]. We distinguish between two kinds of collective decisions: those that are characterized by the decentralization of the process through which individuals reach a decision, and those that involve a step of information centralization, like voting. In this abstract, we focus only on the first kind.

Examples of collective decisions exist across species of very different complexity levels, from insects to humans. A well-known example of a collective decision in insects is the selection of the shorter of two paths between two areas by ants [3]. In humans, an example of a decentralized collective decision is the agreement on the cruise speed on a highway (sometimes different than the established speed limit).

The mechanisms that make collective decision-making possible are very different, for instance, ants use pheromones, bees use waggle dances, humans may use language. Yet, despite the differences, there are some basic principles that underlie these mechanisms, such as the apparent tendency to do or to agree with what others do or think. There is even an adage suggesting that doing what others do is preferable to try and err: "When in Rome, do as the Romans do." In fact, it has been shown that copying others is beneficial in a wide range of environmental conditions [8]. Thus, we believe that it is worthwhile to explore the idea of exploiting the information contained in the actions and communications of others for collective decision-making.

Accordingly, we proposed two models of collective decision-making and tested them in a robotics application [6, 7]. Here, we describe these two models and discuss some of their features from a reinforcement learning point of view. In particular, we highlight the role that the group plays in reinforcing a particular behavior on individuals, what we refer to as *social reinforcement*, that gives the group the ability to make an optimal decision.

2 Models

Our first model is based on an opinion dynamics model that uses the majority rule to integrate the information that different individuals possess [5]. In this model, a population of agents each of which can assume one of two states (also called opinions) evolves as follows: First, a group of three randomly chosen agents is formed. Then, the state of the majority within the group is determined. Finally, all the members of the group adopt the state of he majority (thus, each time a group is formed, a maximum of one agent changes state) and the process is repeated. Regardless of the initial conditions, all the participating agents end up having the same state, that is, this model produces consensus on one of the two available states. The final state depends on the initial density of states: if more than 50% of the agents have one of the two states (say A, for example), then all the agents end up with state A. If exactly 50% of the agents are in state A, then the final state is either A or B with 0.5 probability.

We saw the potential of the dynamics of Krapivsky and Redner's model [5] as a collective decision-making mechanism and tried it in a robotics scenario [6]. However, a robotics application imposes a few constraints that have a big impact on the system's dynamics (as shown in Section 3). First, we interpret agents as robots, which gives them embodiment and situatedness, causing the environment to affect the system's dynamics. Second, states are interpreted as actions that the robots have to repeatedly execute while solving a task. Finally, we explicitly include the passage of time associated with action execution. In our model, the duration of an action is finite but stochastic, that is, two executions of the same action take different completion times as is usually the case in real life.

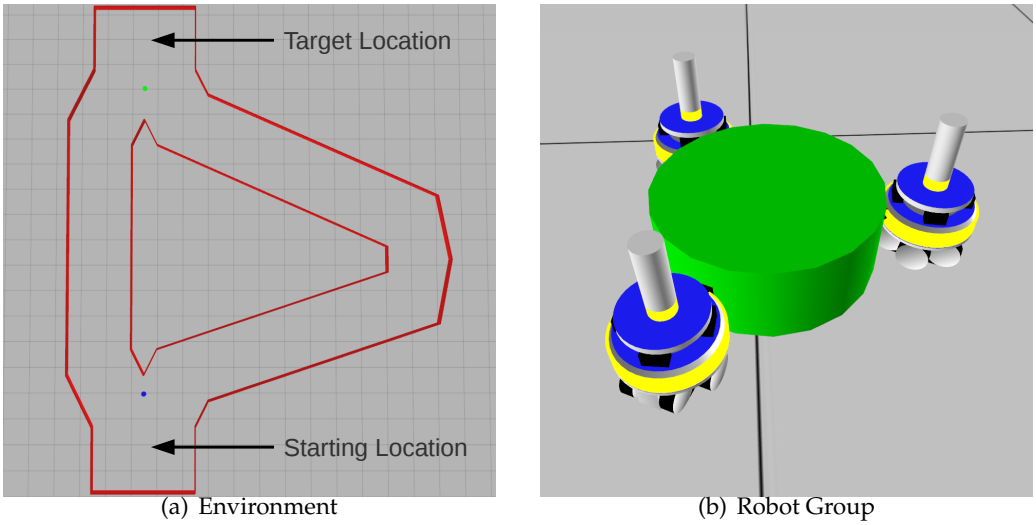
As a test scenario of the system, we used a setting similar to the one used by Goss *et al.* [3] to show that ants can find shortest paths (see Figure 1). The robots' task is to transport heavy objects from one room (starting location) to another (goal location) by forming teams, and going back and forth between these locations. Each time the robots are at the starting location, new teams are formed at random, which allows the mixing of the robots' states.

In our second model, no groups of agents are formed. Instead, individual agents observe the actions performed by other agents. After observation, each agent increases the tendency of performing the observed action. Let the binary variable $X_i \in \{0, 1\}$ to represent an agent i's state. This variable is in turn governed by an internal real-valued variable S_i and a threshold θ. The variable S_i can be thought of as the tendency of agent i to be in one of the two possible states (hereafter, we refer to S_i simply as agent i's tendency). The threshold θ is constant and common to all agents, while S_i is variable and private to each agent.

At each time step t of the system's evolution, an agent i might be able to observe the state of another random agent $j \neq i$. When an agent observes the state of another agent, the observing agent updates its tendency as follows:

$$S_i^{t+1} = (1 - \alpha)S_i^t + \alpha X_j^t \,, \tag{1}$$

where $\alpha \in [0, 1]$ determines how much importance is given to the agent's latest observation (X_j^t) as opposed to the agent's accumulated experience (S_i^t). If α is equal to zero, an agent does not change its tendency to imitate other agents; if α is



(a) Environment (b) Robot Group

Figure 1: Test Scenario. The arena, shown in (a), is a double bridge whose branches differ in length. A group of robots attached to an object is shown in (b). The robots' mission is to transport objects (too heavy for individual robots to move) from the starting to the target location of the arena. The choice robots must make is to take either the longer or the shorter path.

equal to one, an agent copies whatever action another agent performs. After updating its tendency, an agent updates its state as follows:

$$X_i^{t+1} = \begin{cases} 1, & \text{if } S_i^{t+1} > \theta \\ 0, & \text{if } S_i^{t+1} < 1-\theta \\ X_i^t, & \text{if } 1-\theta \leq S_i^{t+1} \leq \theta\,, \end{cases} \tag{2}$$

where $\theta \geq \frac{1}{2}$ due to the symmetry of the actual threshold value that triggers the adoption of one or another state. Thus, an agent's state is a function of its tendency and the threshold θ.

Note how the rule that each individual uses to determine whether to perform a commonly observed action is similar to the basic exponential smoothing equation used for data filtering and time series forecasting [2, 4]. For this reason, we refer to this model as the exponential smoothing model.

3 Simulations

We performed a number of simulations in order to observe the models' dynamics. A summary of the setups and results is presented next.

Majority Rule Model. The duration of actions associated with states A and B are modeled as two normally distributed random variables with means μ_A and μ_B, and standard deviations σ_A and σ_B, respectively. Their ratio $r = \mu_B/\mu_A$ gives a measure of the difference between action execution times and is referred to as latency ratio. Figure 2 shows the dynamics of the majority rule model with a population of 900 agents and 200 teams. When $r \neq 1$, the system achieves consensus on the action of shorter duration even if initially only a minority has a state associated with it. This is seen by the lower critical initial fraction at which the probability of consensus on state A increases from practically zero to almost one.

Exponential Smoothing Model. As in the previous experiments, action durations are normally distributed. In Figure 3, we show the evolution of the average tendency in a population of 100 agents with $r = 2$. When the average duration of the actions associated with each of the two states is equal ($r = 1$) and $S_i^0 = 0.5$, that is, the population of agents will reach a consensus on any of the two states with equal probability (case not shown).

When $r = 2$, the duration of the actions is sufficiently different to induce a strong bias toward the state associated with the action with shorter average execution time even for $\alpha = 0.5$. However, a large value of α means that agents copy any observed state, which produces agents to switch too fast which leads the system to consensus on any of the two states with equal probability.

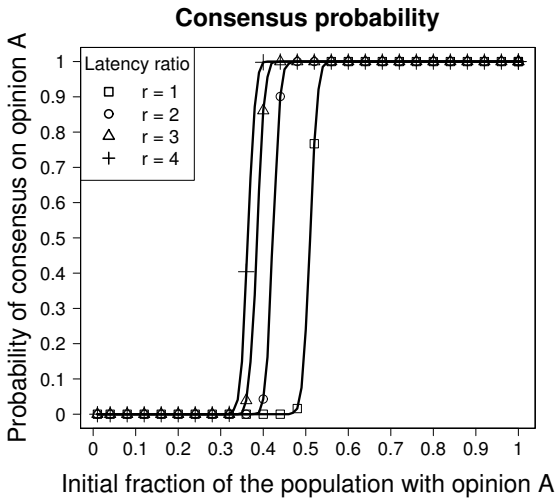


Figure 2: Dynamics of the majority-rule opinion formation model with normally distributed action durations with a population of 900 agents. Results obtained through 1,000 independent runs of a Monte Carlo simulation.

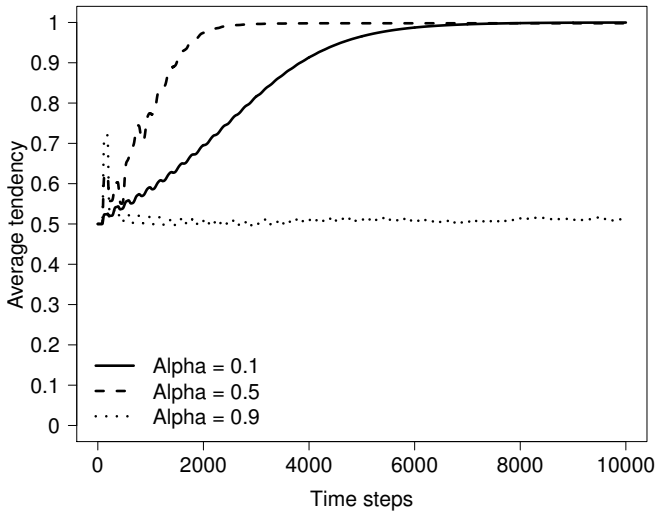


Figure 3: Average tendency over the whole population (100 agents). In these simulations, $\mu_A = 100$, $\sigma_A = \sigma_B = 10$ time steps. The acceptance threshold θ is equal to 0.6 in all cases. Averages obtained through 500 independent runs of a Monte Carlo Simulation.

4 Social Reinforcement

The rules that govern the behavior of the agents in the models presented in the previous section encode the idea of social reinforcement which we understand as the indirect transfer of environmental feedback to a focal individual through the behavior of others. In the following, we discuss how social reinforcement occurs in each of the presented models and its role in the collective decision process.

4.1 Many-to-One Reinforcement

In our first model, robots form groups of three members before executing an action. The action chosen by the group corresponds to the one advocated by the local majority. A robot with a state different from that of the majority is forced to discard it and adopt the state of its peers. For this agent, the state of the majority represents the social reinforcement

because the fact that two other agents share the same state represents information beyond the actual state value. In our setting, since the agents that choose the shorter path return to the starting location more often than the agents that choose the longer path, it is more likely to form groups whose majority is associated with the shorter path. Therefore, the length of the path is encoded in the probability of forming groups of robots with two or more robots advocating for the shorter path. In other words, the state of the environment is contained in the group and therefore, environmental feedback occurs only socially because agents do not measure time or lengths.

4.2 One-to-One Reinforcement

In this model, the signal used to update an agent's tendency is the observed behavior of another agent. If the frequency with which a certain behavior is increased, the the observing agent will increase its tendency to perform that behavior. This is in line with traditional Hebbian learning where repeated stimulation increases the strength of the association between action and stimulus. As with the majority model, if the frequency of observation is linked with environmental interactions, then social reinforcement becomes an indirect feedback channel between an agent and the environment.

5 Conclusions

Animals that form groups to address problems that are too difficult, or even impossible, for individuals to solve have to agree with their peers which actions to take. Upon agreement, we can say that a collective decision has been made. These decisions range from travel directions, to task allocation, to shelter selection. It is important to note that when collective decisions are made, the group is the problem solver, not the isolated individuals. Some researchers call this phenomenon *swarm intelligence*. The models presented in this communication are loosely inspired by natural phenomena, but do not model any real collective decision making mechanism used by animals (at least to best of our knowledge). This not to say that these models do not provide any insights into the dynamics of collective decision making. On the contrary, by reproducing the collective decision making ability of ants without simulating pheromones, our results show that there is a clear distinction between mechanisms and principles and that different mechanisms using the same principles can lead to the same collective-level behavior.

The models presented here illustrate the idea of social reinforcement, that is, the indirect transfer of environmental feedback to a focal individual through the behavior of others. In the majority rule model, social reinforcement occurs through the adoption of the state held by the local majority of the groups formed by robots. In the exponential smoothing model, social reinforcement occurs when a focal agent observes the action of another agent and updates its tendency accordingly. In both cases, when the agents' states are associated with actions with similar outcomes but that take time to perform, the population reaches a consensus on the action that takes less time to perform. In the test scenario used in our experiments, this result is translated into a group selecting the shorter of two paths between two locations without directly measuring time or distance. In these experiments, social reinforcement filters out suboptimal actions.

References

[1] L. Conradt and T. J. Roper. Consensus decision making in animals. *Trends in Ecology & Evolution*, 20(8):449–456, 2005.

[2] E. S. Gardner Jr. Exponential smoothing: The state of the art–Part II. *International Journal of Forecasting*, 22(4):637–666, 2006.

[3] S. Goss, S. Aron, J.-L. Deneubourg, and J.-M. Pasteels. Self-organized Shortcuts in the Argentine Ant. *Naturwissenschaften*, 76(12):579–581, 1989.

[4] R. Hyndman, A. Koehler, K. Ord, and R. Snyder. *Forecasting with Exponential Smoothing. The State Space Approach*. Springer, Berlin, Germany, 2008.

[5] P. L. Krapivsky and S. Redner. Dynamics of Majority Rule in Two-State Interacting Spin Systems. *Physical Review Letters*, 90(23):238701, 4 pages, 2003.

[6] M. A. Montes de Oca, E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, and M. Dorigo. Majority-Rule Opinion Dynamics with Differential Latency: A Mechanism for Self-Organized Collective Decision-Making. *Swarm Intelligence*, 5(3-4):305–327, 2011.

[7] M. A. Montes de Oca, E. Ferrante, A. Scheidler, and L. F. Rossi. Binary Consensus via Exponential Smoothing. In R. Colbaugh et al., editors, *Proceedings of the Second International Conference on Complex Sciences: Theory and Applications (COMPLEX 2012)*. 2012. To appear.

[8] L. Rendell, R. Boyd, D. Cownden, M. Enquist, K. Eriksson, M. W. Feldman, L. Fogarty, S. Ghirlanda, T. Lillicrap, and K. N. Laland. Why Copy Others? Insights from the Social Learning Strategies Tournament. *Science*, 328(5975):208 – 213, 2010.

Strategic Robot Learner for Interactive Goal-Babbling

Sao Mai Nguyen *
Flowers Team
INRIA
Bordeaux, France
nguyensmai@gmail.com

Pierre-Yves Oudeyer †
Flowers Team
INRIA
Bordeaux, France pierre-yves.oudeyer@inria.fr

Abstract

The challenges posed by robots operating in human environments on a daily basis and on the long-term point out the importance of adaptivity to changes which can be unforeseen at design time. Therefore, the robot must learn continuously in an open-ended, non-stationary and high dimensional space. It can not possibly explore all its environment to learn about everything within a life-time. We propose to investigate the relationship between two classical learning modes: imitation learning and intrinsically-motivated autonomous exploration. We buildan algorithmic architecture where relationships between the two sampling modes intertwine into a hierarchical structure, called Socially Guided Intrinsic Motivation with Active Choice of Teachers and Strategies (SGIM-ACTS).

Indeed, we have built an intrinsically motivated active learner which learns how its actions can produce varied consequences or outcomes. For instance, the robot learns to throw a ball at different distances, by associating a distance (outcome) to a specific movement (action). It actively learns online by sampling data which it chooses by using several sampling modes. On the meta-level, it actively learns which data collection strategy is most efficient for improving its competence and generalising from its experience to a wide variety of outcomes. The interactive learner thus learns multiple tasks in a structured manner, discovering by itself developmental sequences.

We contribute to different fields of machine learning:

- imitation learning : we propose a unified structure to address simulateneously the fundamental questions of imitation learning: what, how, when and who to imitate. In particular in interactive learning,we identify advantages of combining autonomous exploration and socially guided exploration, and build an agent which decides by itself when to interact with teachers.
- multi-task learning : SGIM-ACTS can discover the structure of its environment by a goal-oriented exploration. We propose a unified architecture to approach goal-oriented imitation learning (to reproduce a demonstrated goal) and goal-directed autonomous exploration (goals guiding policy exploration).
- active learning : we investigate different levels of active learning : the learner can decide which action to take, or which goal to aim, or which sampling mode to use. Its decisions are made online, driven by artificial curiosity based on its monitoring of learning progress.
- hierarchical learning : we propose a hierarchical learning architecture to learn on several levels: policy, outcome, and mode spaces. The learner relies on hierarchical active decisions of what and how to learn driven by empirical evaluation of learning progress for each sampling mode on a meta-level.

Keywords: active learning, interactive learning, imitation learning, goal-oriented exploration, data-collection, exploration, programming by demonstration

Acknowledgements

This work was supported by the French ANR program (ANR 2010 BLAN 0216 01) through Project MACSi, as well by ERC Starting Grant EXPLORERS 240007.

*http://nguyensmai.free.fr
†www.pyoudeyer.com

1 Strategic Active Learning for Life-Long Acquisition of Multiple Skills

Life-long learning by robots to acquire multiple skills in unstructured environments poses challenges of learning in large and high-dimensional sensorimotor spaces, while their life-time allows only limited number of collected data.

1.1 Active Learning for Producing Varied Outcomes with Multiple sampling modes

The choice of a sampling mode can be formalised under the notion of strategic learning [11]. One perspective is learning to achieve varied outcomes and aims at selecting which outcome to spend time on. Another perspective is learning how to learn, by making explicit the choice and dependence of the learning performance on the method. However most studies have not addressed the learning of both how to learn and what to learn, to select at the same time which outcome to spend time on, and which learning method to use. Only [11] studies the framework of these questions. In initial work to address learning for varied outcomes with multiple methods, we proposed in [15] the Socially Guided Intrinsic Motivation by Demonstration (SGIM-D) algorithm which uses both 1) socially guided exploration, especially programming by demonstration [3] and 2) intrinsically motivated exploration, which are active learning algorithms based on measures of the evolution of the learning performance [16] to reach goals in a continuous outcome space.

In this paper, we extend this work and study how a learning agent can achieve varied outcomes in structured continuous outcome spaces, and how he can learn which sampling mode to choose among 1) active self-exploration, 2) reproduction of the demonstrated outcome or *emulation* of a teacher actively selected among available teachers, 3) reproduction of the demonstrated policy or *mimicry* of an actively selected teacher.

1.2 Actively Learning When, Who and What to Imitate

In this paper, we develop our social guidance into interactive learning. The learner actively requests for the information it needs and when it needs help [6]. For the model and experiments presented below, our agent learns to answer the four main questions of imitation learning: "what, how, when and who to imitate" [9, 4] at the same time. We address active learning for varied outcomes with multiple sampling mode, multiple teachers, with a structured continuous outcome space (embedding sub-spaces with different properties). The sampling modes we consider are autonomous self-exploration, emulation and mimicking, by interactive learning with several teachers.

1.3 Our Approach

Let us consider an agent learning motor skills, i.e. how to induce any outcome $\mathcal{A} \in \mathbb{A}$ with motor programs $\pi \in \mathbb{P}$. We parameterise the outcome space with parameters $a \in A$. A policy π_b is described by motor primitives parameterised by $b \in B$.The probability of that the policy parameter b produces the outcome of parameter a is $\tilde{p}(a|b,c)$, where the probability density $\tilde{p}$ represents the physics of the environment which the agent estimates. The association (b,a) corresponds to a learning exemplar that will be memorised.

To solve the problem formalised above, we propose a system, called Socially Guided Intrinsic Motivation with Active Choice of Teacher and Strategy (**SGIM-ACTS**) that allows an online interactive learning of inverse models in continuous high-dimensional robotic sensorimotor spaces with multiple teachers, and sampling mode. SGIM-ACTS learns various outcomes with different types of outcomes, and generalises from sampled data to continuous sets of outcomes.

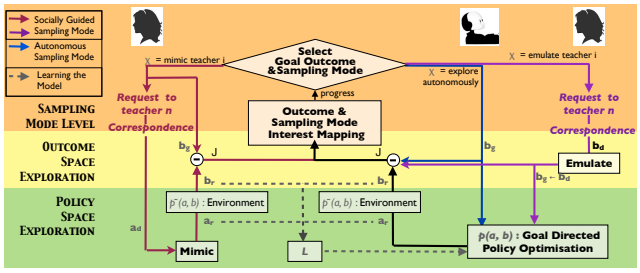
Technically, we adopt a method of generalisation of policies for new outcomes similar to [10, 8], except that instead of using a pool of examples given by the teacher preset from the beginning of the experiment to learn outcomes specified by the engineer of the robot, the SGIM-ACTS algorithm decides by itself which outcomes it needs to learn more to better generalise for the whole outcome space, like in [2]. Moreover, SGIM-ACTS actively requests the teacher's demonstrations online, by choosing online a good sampling mode, similarly to [1], except that we instead of a discrete, we use a continuous outcome space. SGIM-ACTS also interacts with several teachers and uses several social learning methods.

Our active learning approach is inspired by 1) psychological theories for socially guided learning [5], 2) teleological learning [7] which considers actions as goal-oriented, and 3) intrinsic motivation in psychology [17] which triggers spontaneous exploration and curiosity in humans, which recently led to novel robotic and machine active learning methods which outperform traditional active learning methods [2].

After this definition of the problem addressed in this paper, we describe the design of our **SGIM-ACTS** (Socially Guided Intrinsic Motivation with Active Choice of Teacher and Strategy) algorithm. Then we show through an illustration experiment that SGIM-ACTS efficiently learns to realise different types of outcomes in continuous outcome spaces, and it coherently selects the right teacher to learn from.

Algorithm 2.1 SGIM-ACTS

Input: the different modes $\chi_\alpha, ...\chi_\kappa$.
Initialization: partition of outcome space $\mathcal{R} \leftarrow$ singleton A
Initialization: episodic memory (collection of produced outcomes) $\mathcal{H} \leftarrow$ empty memory
Initialization: $e \leftarrow 1$
loop
  $a_i, \chi \leftarrow$ Select Goal Outcome and Strategy$(\mathcal{R})$
  if $\chi =$ Mimic teacher i mode **then**
    $(\zeta_d, a_d) \leftarrow$ ask and observe demonstration to teacher i.
    $\gamma_1 \leftarrow$ Competence for a_g
    $\mathcal{D}_e \leftarrow$ Mimic Action(ζ_d)
    $p_{e+1} \leftarrow \mathcal{L}(p_e, \mathcal{D}_e)$
  else if $\chi =$ Emulate teacher i mode **then**
    $(\zeta_d, a_d) \leftarrow$ ask and observe demonstration to teacher i.
    Emulation: $a_g \leftarrow a_d$
    $\gamma_1 \leftarrow$ Competence for a_g
    $\mathcal{D}_e \leftarrow$ Goal-Directed Policy Optimisation(a_g)
    $p_{e+1} \leftarrow \mathcal{L}(p_e, \mathcal{D}_e)$
  else
    $\chi =$ **Intrinsic Motivation mode**
    $a_g \leftarrow a_i$
    $\gamma_1 \leftarrow$ Competence for a_g
    $\mathcal{D}_e \leftarrow$ Goal-Directed Policy Optimisation(a_g)
    $p_{e+1} \leftarrow \mathcal{L}(p_e, \mathcal{D}_e)$
  end if
  $\gamma_2 \leftarrow$ Competence for a_g
  $nbA \leftarrow$ number of episodes in $\mathcal{D}_e$
  $prog \leftarrow 2(sig(\alpha_p * \frac{\gamma_2 - \gamma_1}{|T_i| \cdot nbA}) - 1)$
  Append $\mathcal{D}_e$ to $\mathcal{H}$
  $\mathcal{R} \leftarrow$ Update Outcome and Strategy Interest Mapping$(\mathcal{R}, \mathcal{H}, a_g, prog, \chi)$
  $e \leftarrow e + 1$
end loop



(a) Time flow chart of SGIM-ACTS, which combines Intrinsic Motivation and Mimicking and Emulation into 3 layers that pertain to the sampling mode, the outcome space and the policy space exploration respectively.

Figure 1: SGIM-ACTS algorithm.

2 Algorithm Description

In this section, we describe the SGIM-ACTS architecture by giving a behavioural outline (Algorithm 2.1 and fig. 1a).

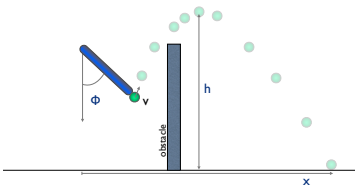
SGIM-ACTS is an architecture that merges intrinsically motivated self-exploration with interactive learning as socially guided exploration. In the latter case, a teacher performs an observed trajectory ζ which achieves an observed outcome b_d (intentional or unintentional). SGIM-ACTS learns by episodes during which it actively chooses simultaneously an outcome $b_g \in T$ to reach and a sampling mode with a specific teacher. Its choice χ is selected among : intrinsically motivated exploration, mimicry from teacher 1, emulation of teacher 1, mimicry from teacher 2, emulation of teacher 2 .... (fig. 1a).

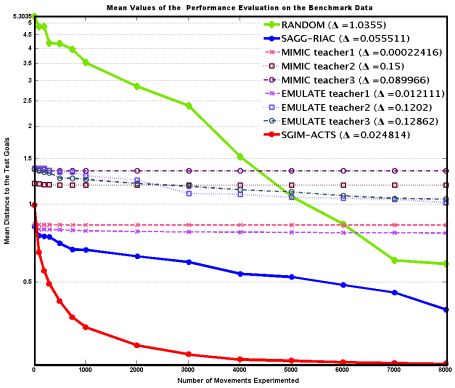
In an episode under a mimicking mode, our SGIM-ACTS learner actively self-generates a goal b_g where its competence improvement is maximal. The SGIM-ACTS learner explores preferentially goal outcomes easy to reach and where it makes progress the fastest. The selected teacher answers its request with a demonstration $[\zeta_d, b_d]$ to produce an outcome b_d that is closest to b_g. The robot mimics the teacher to reproduce ζ_d, for a fixed duration, by performing policies a_θ which are small variations of an approximation of ζ_d. In an episode under an emulation mode, our SGIM-ACTS learner observes from the selected teacher a demonstration $[\zeta_d, b_d]$. It tries different policies using goal-directed optimisation algorithms to approach the observed outcome b_d, without taking into account the demonstrated policy ζ_d. It re-uses and optimises its policy repertoire built through its past autonomous and socially guided explorations. The episode ends after a fixed duration. In an episode under the intrinsic motivation mode, it explores autonomously following the SAGG-RIAC algorithm [2]. It actively self-generates a goal b_g where its competence improvement is maximal, as in the mimicking mode. Then, it explores which policy a_θ can achieve b_g best. It tries different policies to approach the self-determined outcome b_g, as in the emulation mode. The episode ends after a fixed duration. The intrinsic motivation and emulation mode differ mainly by the way the goal outcome is chosen. The details of this **3-layered (policy, outcome and mode space explorations) hierarchical architecture** and the different functions and levels can be read in [12, 13].

3 Experiment

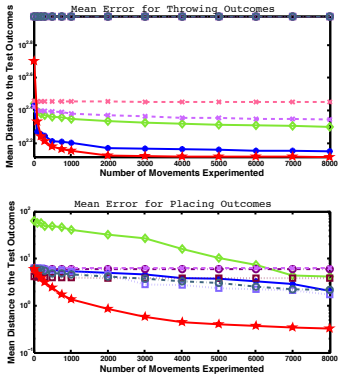
3.1 Experimental setup

Figure 2: An arm, described by its angle ϕ, is controlled by a motor primitive with 14 continuous parameters (taking bounded values) that determine the evolution of its acceleration $\ddot{\phi}$. A ball is held by the arm and then released at the end of the motion. The objective of the robot is to learn the mapping between the parameters of the motor primitive and two types of outcomes he can produce: a ball thrown at distance x and height h, or a ball placed at the arm tip at angle ϕ with velocity smaller than $|v_{max}|$.



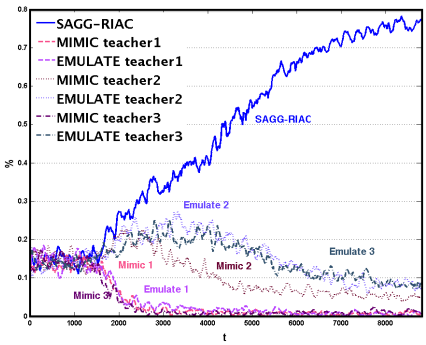


(a) Mean error for the different learning algorithms averaged over the two sub outcome spaces (final variance value Δ is indicated in the legend) .

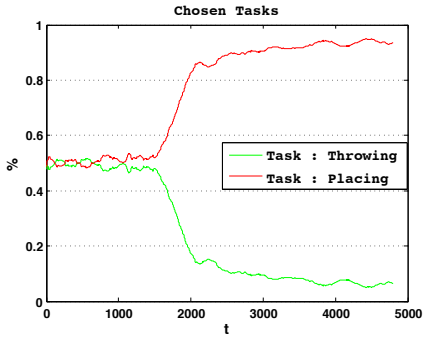


(b) Mean error for the different learning algorithms for each of the throwing outcomes and placing outcomes separately. The legend is the same as in fig. 3a.

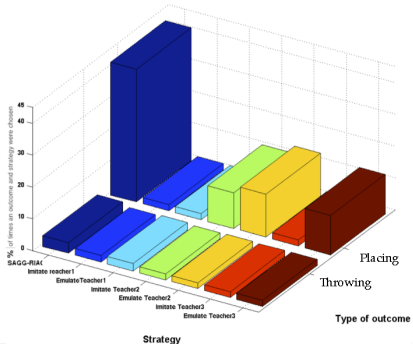
Figure 3: Error plots.



(a) Strategy chosen by SGIM-ACTS through time: percentage of times each sampling mode is chosen for several runs of the experiment.



(b) Outcome chosen by SGIM-ACTS through time: percentage of times each kind of outcome is chosen for several runs of the experiment.



(c) Consistency in the choice of outcome, teacher and mode: percentage of times each sampling mode, teacher and outcome are chosen over all the history of the robot.

Figure 4: sampling modes chosen.

We illustrate in the following section this hierarchical algorithm through a simulation where a robot learns to throw a ball or to place it at different angles (fig. 2) with 7 sampling modes: intrinsically motivated exploration, mimicry from 3 teachers and emulation from 3 teachers. The 3 teachers considered are respectively an expert in throwing balls, an expert in placing balls, and an expert in placing balls with correspondence problems. We prepared demonstration sets for each teacher, so that the demonstrated outcomes are equally distributed in the reachable space. A demonstration is stored as a pair or policy and outcome parameters. When a teacher is requested a demonstration for emulation, he gives a random demonstration among its demonstration set. The details of the experimental setup can be read in [13]. In the next section, we present the results of the experiment.

3.2 Results

We compared SGIM-ACTS with 4 other learning algorithms: random exploration of the policy space, SAGG-RIAC [2], mimicry and emulation. Fig. 3a shows that SGIM-ACTS decreases its cumulative error for both placing and throwing. It performs better than autonomous exploration by random search or intrinsic motivation, and better than mimicry or emulation with any teacher. Fig. 3b shows that SGIM-ACTS error rate for both placing and throwing is low. For throwing, SGIM-ACTS performs the best in terms of error rate and speed because it could find the right mode. While mimicking and emulating teacher 1 decreases the error as expected, mimicking and emulating a teacher who is expert in another kind of outcomes and is bad in that outcome leaves a high error rate. For placing, SGIM-ACTS makes less error than all other algorithms. Indeed, as we expected, mimicking the teacher 2, and emulating teachers 2 and 3 enhances low error rates, while mimicking a teacher with correspondence problem (teacher 3) or an expert on another outcome (teacher 1) gives poor result. We also note that for both outcomes, mimicry does not lead to important learning progress, and the error curve is almost flat. This is due to the lack of exploration which leads the learner to ask demonstrations for outcomes only in a small subspace.

Indeed, we see in fig. 4a which illustrates the percentage times each sampling mode is chosen by SGIM-ACTS with respect to time, that mimicry of teacher 3, which lacks efficiency because of the correspondence problem, is seldom chosen by SGIM-ACTS. Mimicry and emulation of teacher 1 is also little used because autonomous learning learns

quickly throwing outcomes. Teachers 2 and 3 are exactly the same with respect to the outcomes they demonstrate, and are emulated in the same proportion. This figure also shows that the more the learner cumulates knowledge, the more autonomous he grows : his percentage of autonomous learning increases steadily.

Not only does he choose the right sampling mode, but also the right outcome to concentrate on. Fig. 4b shows that he concentrates in the end more on placing, which are more difficult.

Finally, fig. 4c shows the percentage of times over all the experiments where he chooses at the same time each outcome type, a sampling mode and a teacher. We can see that for the placing outcomes, he seldom requests help from the teacher 1, as he learns that teacher 1 does not know how to place the ball. Likewise, because of the correspondence problems, he does not mimic teacher 3. But he learns that mimicking teacher 2 and emulating teachers 2 and 3 are useful for placing outcomes. For the throwing outcomes, he uses slightly more the autonomous exploration sampling modes, as he can learn efficiently by himself. The high percentage for the other sampling mode is due to the fact that the throwing outcomes are easy to learn, therefore are learned in the beginning when a lot of sampling of all possible sampling modes is carried out. SGIM-ACTS is therefore consistent in its choice of outcomes , sampling modes and teachers.

4 Conclusion and Discussion

We presented the **SGIM-ACTS** (Socially Guided Intrinsic Motivation with Active Choice of Teacher and Strategy) algorithm that efficiently and actively combines autonomous self-exploration and interactive learning, to address the learning of multiple outcomes, with outcomes of different types, and with different sampling modes. In particular, it learns actively to decide on the fundamental questions of programming by demonstration: *what and how* to learn; but also *what, how, when and who* to imitate. This interactive learner decides efficiently and coherently whether to use social guidance. It learns when to ask for demonstration, what kind of demonstrations (action to mimic or outcome to emulate) and who to ask for demonstrations, among the available teachers. Its hierarchical architecture bears three levels. The lower level explores the policy parameters space to build skills for determined goal outcomes. The upper level explores the outcome space to evaluate for which outcomes he makes the best progress. A meta-level actively chooses the outcome and sampling mode that leads to the best competence progress. We showed that SGIM-ACTS can focus on the outcome where it learns the most, while choosing the most appropriate associated sampling mode. The active learner can explore efficiently a composite and continuous outcome space to be able to generalise for new outcomes of the outcome spaces.

Even in the case of correspondence problems, the system still takes advantage of the demonstrations to bias its exploration of the outcome space, as argued in [14]. Future work should test SGIM-ACTS on more complex environments, and with real physical robots and everyday human users. It would also be interesting to compare the outcomes selected by our system to developmental behavioural studies, and highlight developmental trajectories.

References

[1] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *The Journal of Machine Learning Research,*, 5:255–291, 2004.

[2] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

[3] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. *Handbook of Robotics*, chapter Robot Programming by Demonstration. Number 59. MIT Press, 2007.

[4] Cynthia Breazeal and B. Scassellati. Robots that imitate humans. *Trends in Cognitive Sciences*, 6(11):481–487, 2002.

[5] J. Call and M. Carpenter. *Imitation in animals and artifacts*, chapter Three sources of information in social learning, pages 211–228. Cambridge, MA: MIT Press., 2002.

[6] Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34, 2009.

[7] Gergely Csibra. Teleological and referential understanding of action in infancy. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):447, 2003.

[8] B.C. da Silva, G. Konidaris, and Andrew G. Barto. Learning parameterized skills. In *29th International Conference on Machine Learning (ICML 2012)*, 2012.

[9] Kerstin Dautenhahn and Chrystopher L. Nehaniv. *Imitation in Animals and Artifacts*. MIT Press, 2002.

[10] Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, pages 1–19, 2012. 10.1007/s10514-012-9290-3.

[11] Manuel Lopes and Pierre-Yves Oudeyer. The Strategic Student Approach for Life-Long Exploration and Learning. In *IEEE Conference on Development and Learning / EpiRob*, San Diego, États-Unis, November 2012.

[12] Sao Mai Nguyen, Serena Ivaldi, Natalia Lyubova, Alain Droniou, Damien Gerardeaux-Viret, David Filliat, Vincent Padois, Olivier Sigaud, and Pierre-Yves Oudeyer. Learning to recognize objects through curiosity-driven manipulation with the icub humanoid robot. In *IEEE International Conference on Development and Learning - Epirob*, 2013.

[13] Sao Mai Nguyen and Pierre-Yves Oudeyer. Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. *Paladyn Journal of Behavioural Robotics*, 3(3):136–146, 2012.

[14] Sao Mai Nguyen and Pierre-Yves Oudeyer. Properties for efficient demonstrations to a socially guided intrinsically motivated learner. In *21st IEEE International Symposium on Robot and Human Interactive Communication*, 2012.

[15] Sao Mai Nguyen and Pierre-Yves Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, pages 1–22, 2013.

[16] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 2007.

[17] Richard M. Ryan and Edward L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 25(1):54 – 67, 2000.

Learning how to reach various goals by autonomous interaction with the environment: unification and comparison of exploration strategies

Clément Moulin-Frier
Flowers team, Inria / ENSTA-Paristech
Bordeaux, France
clement.moulin-frier@inria.fr

Pierre-Yves Oudeyer
Flowers team, Inria / ENSTA-Paristech
Bordeaux, France
pierre-yves.oudeyer@inria.fr

Abstract

In the field of developmental robotics, we are particularly interested in the exploration strategies which can drive an agent to learn how to reach a wide variety of goals. In this paper, we unify and compare such strategies, recently shown to be efficient to learn complex non-linear redundant sensorimotor mappings. They combine two main principles. The first one concerns the space in which the learning agent chooses points to explore (motor space vs. goal space). Previous works (Rolf et al., 2010; Baranes and Oudeyer, 2012) have shown that learning redundant inverse models could be achieved more efficiently if exploration was driven by goal babbling, triggering reaching, rather than direct motor babbling. Goal babbling is especially efficient to learn highly redundant mappings (e.g the inverse kinematics of a arm). At each time step, the agent chooses a goal in a goal space (e.g uniformly), uses the current knowledge of an inverse model to infer a motor command to reach that goal, observes the corresponding consequence and updates its inverse model according to this new experience. This exploration strategy allows the agent to cover the goal space more efficiently, avoiding to waste time in redundant parts of the sensorimotor space (e.g executing many motor commands that actually reach the same goal). The second principle comes from the field of active learning, where exploration strategies are conceived as an optimization process. Samples in the input space (i.e motor space) are collected in order to minimize a given property of the learning process, e.g the uncertainty (Cohn et al., 1996) or the prediction error (Thrun, 1995) of the model. This allows the agent to focus on parts of the sensorimotor space in which exploration is supposed to improve the quality of the model.

This paper shows how an integrating probabilistic framework allows to model several recent algorithmic architectures for exploration based on these two principles, and compare the efficiency of various exploration strategies to learn how to uniformly cover a goal space.

Keywords: Exploration strategies, goal babbling, active learning, developmental robotics.

Acknowledgements

This work was partially financed by ERC Starting Grant EXPLORERS 240 007.

1 Introduction

The learning of sensorimotor tasks, for example reaching objects with the hand or controlling the shape of a vocal tract to produce particular sounds, involves the learning of complex sensorimotor mappings. This latter generally requires to build a model of the relationships between parts of the sensorimotor space. For example, one might need to predict the positions of the hand knowing the joint configurations, or to control the shape the vocal tract to produce the sound of particular words.

Let us introduce the problem more formally. A learning agent interacts with a surrounding environment through motor commands M and sensory perceptions S. We call $f : M \rightarrow S$ the unknown function defining the physical properties of the environment, such that when the agent produces a motor command $m \in M$, it then perceives $s \in S$. Classical robotic problems are e.g. the prediction of the sensory effect of an intended motor command through a forward model $\tilde{f} : M \rightarrow S$, or the control of the motor system to reach sensory goals through an inverse model $\tilde{f^{-1}} : S \rightarrow M$. The agent has to learn such models by collecting (m, s) pairs through its interaction with the environment, i.e. by producing $m \in M$ and observing $s = f(m)$. These learning processes are often difficult for several reasons: 1) the agent has to deal with uncertainties both in the environment and in its own sensorimotor loop, 2) M and S can be highly dimensional, such that random sampling in M to collect (m, s) pairs can be a long and fastidious process, 3) f can be strongly non-linear, such that the learning of $\tilde{f}$ from experience is not trivial, 4) f can be redundant (many M to one S), such that the learning of $\tilde{f^{-1}}$ is an ill-posed problem (f^-1 does not exist, or cannot be directly recovered from f).

When a learning process faces these issues, random motor exploration (or motor babbling) in M is not a realist exploration strategy to collect (m, s) pairs. Due to high dimensionality, data are precious whereas, due to non-linearity and/or redundancy, data are not equally useful to learn an adequate forward or inverse model.

2 Exploration strategies

Computational studies have shown the importance of developmental mechanisms guiding exploration and learning in high-dimensional M and S spaces and with highly redundant and non-linear f (Oudeyer et al., 2007; Baranes and Oudeyer, 2012). Among these guiding mechanisms, intrinsic motivations, generating spontaneous exploration in humans (Berlyne, 1954; Deci and Ryan, 1985), have been transposed in curiosity-driven learning machines (Schmidhuber, 1991; Barto et al., 2004; Schmidhuber, 2010) and robots (Oudeyer et al., 2007; Baranes and Oudeyer, 2012) and shown to yield highly efficient learning of inverse models in high-dimensional redundant sensorimotor spaces (Baranes and Oudeyer, 2012). Efficient versions of such mechanisms are based on the active choice of learning experiments that maximize learning *progress*, for e.g. improvement of predictions or of competences to reach goals (Schmidhuber, 1991; Oudeyer et al., 2007). This automatically drives the system to explore and learn first easy skills, and then explore skills of progressively increasing complexity.

This led to the implementation of various exploration strategies (Baranes and Oudeyer, 2012), which differ in the way the agent iteratively collects (m, s) pairs to learn forward and/or inverse models (comparing random vs. learning progress based exploration, in either the motor M or the sensory S spaces). These strategies are summarized below (the original name of the corresponding algorithm appears in parenthesis).

- **Random motor exploration (ACTUATOR-RANDOM):** at each time step, the agent randomly chooses an articulatory command $m \in M$, produces it, observes $s = f(m)$ and updates its sensorimotor model according to this new experience (m, s).
- **Random goal exploration (SAGG-RANDOM):** at each time step, the agent randomly chooses a goal $s_g \in S$ and tries to reach it by producing $m \in M$ using an inverse model $\tilde{f^{-1}}$ learned from previous experience. It observes the corresponding sensory consequence $s = f(m)$ and updates its sensorimotor model according to this new experience (m, s).
- **Active motor exploration (ACTUATOR-RIAC):** at each time step, the agent chooses a motor command m by maximizing an interest value in M based on an empirical measure of the learning progress in prediction in its recent experience. The agent uses a forward model $\tilde{f}$ learned from its past experience to make a prediction $s_p \in S$ for the motor command m. It produces m and observe $s = f(m)$. The agent updates its sensorimotor model according to the new experience (m, s). A measure of learning accuracy is computed from the distance between s_p and s, which is used to update the interest model in the neighborhood of m.
- **Active goal exploration (SAGG-RIAC):** at each time step, the agent chooses a goal s_g by maximizing an interest value in S based on an empirical measure of the learning progress in competence to reach goals in its recent experience. It tries to reach s_g by producing $m \in M$ using a learned inverse model $\tilde{f^{-1}}$. It observes the corresponding sensory consequence $s \in S$ and updates its sensorimotor model according to this new experience

(m, s). A measure of learning accuracy is computed from the distance between s_g and s, which is used to update the interest model in the neighborhood of s_g.

In the two active strategies, the measure of interest was obtained by recursively splitting the space (M in ACTUATOR-RIAC, S in SAGG-RIAC) into sub-regions during the agent life. Each region maintains its own empirical measure of learning progress from its learning accuracy history in a relative time window. This accuracy is defined as the opposite of the distance between s_p and s in the active motor strategy, between s_g and s in the active goal one. These active strategies are very related to the field of *active learning*, although this latter often constrains the interest measure to be defined in the input space (M in our formalism).

We have recently suggested to classify these four strategies along two dimensions (Moulin-Frier and Oudeyer, 2013a,b). The first one corresponds to the space X in which the agent drives its exploration, which is here either M (motor strategies) or S (goal strategies). We call it the *choice space*. The second dimension is the kind of interest measure used by this agent at each time step to choose a point in its choice space, either uniform leading to a random sampling in X (random strategies), or based on empirical measurements, here the learning progress in prediction or control (active strategies).

3 Probabilistic modeling

We use a probabilistic framework where the notations are inspired by Jaynes (2003) and Lebeltel et al. (2004). Upper case A denotes a probabilistic variable, defined by its continuous, possibly multidimensional and bounded domain $\mathcal{D}(A)$. The conjunction of two variables $A \wedge B$ can be defined as a new variable C with domain $\mathcal{D}(A) \times \mathcal{D}(B)$. Lower case a will denote a particular value of the domain $\mathcal{D}(A)$. $p(A \mid \omega)$ is the probability distribution over A knowing some preliminary knowledge ω (e.g. the parametric form of the distribution, a learning set ...). Practically, ω will serve as a model identifier, allowing to define different distributions of the same variable, and we will often omit it in the text although it will be useful in the equations. $p(A\ B \mid \omega)$ is the probability distribution over $A \wedge B$. $p(A \mid [B = b]\ \omega)$ is the conditional distribution over A knowing a particular value b of another variable B (also noted $p(A \mid b\ \omega)$ when there is no ambiguity on the variable B). For simplicity, we will often confound a variable and its domain, saying for example *"the probability distribution over the space A"*.

Considering that we know the joint probability distribution over the whole sensorimotor space, $p(M\ S \mid \omega_{SM})$, Bayesian inference provides the way to compute every conditional distribution over $M \wedge S$. In particular, we can compute the conditional distribution over Y knowing a particular value x of X, as long as X and Y correspond to two complementary sub-domains of $M \wedge S$ (i.e. they are disjoint and $X \wedge Y = M \wedge S$). Thus, the prediction of $s_p \in S$ from $m \in M$ in the active motor exploration strategy, or the control of $m \in M$ to reach $s_g \in S$ in the active or random goal exploration strategies, correspond to the probability distributions $p(S \mid M\ \omega_{SM})$ and $P(M \mid S\ \omega_{SM})$, respectively. More generally, whatever the choice and inference spaces X and Y, as long as they are subspaces of $M \wedge S$ and they are disjoint, Bayesian inference allows to compute $p(Y \mid X\ \omega_{SM})$.

Such a probabilistic modeling is also able to express the interest model, that we will call ω_I, such that the agent draws points in the choice space X according to the distribution $p(X \mid \omega_I)$. In the random motor and goal exploration strategies, this distribution is uniform, whereas it is a monotonically increasing function of the empirical interest measure in the case of the active exploration strategies.

Given this probabilistic framework, Algorithm 1 describes our generic exploration algorithm.

Algorithm 1 Generic exploration algorithm

1: set choice space X
2: **while** true **do**
3: $x \sim p(X \mid \omega_I)$
4: $y \sim p(Y \mid x\ \omega_{SM})$
5: $m = M\ ((x, y))$
6: $s = exec(m)$
7: $e = distance(S(x, y), s)$
8: $update(\omega_{SM}, (m, s))$
9: $update(\omega_I, (x, e))$
10: **end while**

Line 1 defines the choice space of the exploration strategy. For example X is set to M for the motor strategies and to S for the goal strategies described in Section 2, but the formalism can also deal with any part of $M \wedge S$ as the choice space. Line 3, the agent draws a point x in the choice space X according to the current state of its interest model ω_I, through the probability distribution $p(X \mid \omega_I)$ encoding the current interest over X. This distribution is uniform in the case of

the random strategies and related to the learning progress in prediction or control in the active strategies of Section 2. Line 4, the agent draws a point y in the inference space Y (remember that Y is such that $X \wedge Y = M \wedge S$) according to the distribution $p(Y \mid x\, \omega_{SM})$, using Bayesian inference on the joint distribution $p(M\, S \mid \omega_{MS})$. If $X = M$, and therefore $Y = S$, this corresponds to a prediction tasks $p(S \mid [M = x])$; if $X = S$, and therefore $Y = M$, this corresponds to a control task $p(M \mid [S = x])$. Line 5, the agent extracts the motor part m of (x, y), noted $M\,((x, y))$, i.e. x if $X = M$, y if $X = S$. Line 6, the agent produces m and observe $s = exec(m)$, i.e. $s = f(m)$ with possible sensorimotor constraints and noises. Line 7 the agent computes a learning error as a distance betwween the sensory part of (x, y), noted $S(x, y)$, i.e. y if $X = M$, x if $X = S$, and the actual sensory consequence s. Line 8 the agent updates its sensorimotor model according to its new experience (m, s). Line 9 the agent updates its interest model according to the choice $x \in X$ it made and the associated learning error e.

In this framework, we are able to more formally express each algorithm presented in Section 2. The random motor strategy (ACTUATOR-RANDOM) is the simpler case where the choice space is $X = M$ and the interest model of line 3 is set to a uniform distribution over X. Inference in line 4 is here useless because motor extraction (line 5) will return the actual choice x and that there is no need to update the interest model in line 9. The active motor strategy (ACTUATOR-RIAC) differs from the previous one by the interest model of line 3 which favors regions of X $(= M)$ maximizing the learning progress in prediction. This latter is computed at the update step of line 9 using the history of previous learning errors computed at line 7, which are here distances between the prediction $y \in Y$ computed on line 4 (with $Y = S$) and the actual realization $s \in S$ of line 6. The random goal strategy (SAGG-RANDOM) is the case where the interest model is uniform and the choice space is S, implying that the inference corresponds to a control task to reach $x \in X$ by producing $y \in Y$ (with $X = S$ and therefore $Y = M$). Finally, the active goal strategy (SAGG-RIAC) differs from the previous one by the interest model which favors regions of X $(= S)$ maximizing the learning progress in control. This latter is computed in the same way that for ACTUATOR-RANDOM, except that the distance is here between the chosen goal $x \in X$ and the actual realization $s \in S$ (with $X = S$).

We do not develop in this abstract how the sensorimotor and the interest distributions can be practically implemented (see e.g. Moulin-Frier and Oudeyer (2013a,b) and further papers of the authors). We therefore directly provide comparative results in the next section, asking the reader to assume that these distributions can be computed in a way or another.

4 Results

In this section, we perform computer simulations with a simulated sensorimotor agent. The motor space M is articulatory (7-dimensional), and the sensory one is auditory (2-dimensional). The unknown function $f : M \to S$ is provided by the articulatory synthesizer of the DIVA model described in Guenther et al. (2006), a computational model of the human vocal tract. We do not present it here, the only important point being that the articulatory-to-auditory transformation is known to be redundant and non-linear. The agent implements Algorithm 1 with different choice spaces and interest distributions corresponding to the four strategies ACTUATOR-RANDOM, ACTUATOR-RIAC, SAGG-RANDOM and SAGG-RIAC described in Section 2. We evaluate the efficiency of the obtained sensorimotor models to achieve a control task, i.e. to reach a test set of goals uniformly distributed in the reachable auditory space.

Figure 1 shows the performance results of the four exploration strategies on a control task during the life time of learning agents. We observe that the strategies with S as the choice space (random and active goal strategies) are significantly more efficient that those with M (random and active motor strategies), i.e. both convergence speed (say around 100 updates) and generalization at the end of the simulation (500 updates) are better. Moreover, both convergence speed and generalization are better for the active than for the random goal strategy. These results are similar (though less significant) to those obtained in previous experiments (Baranes and Oudeyer, 2012) in other sensorimotor spaces (e.g. a arm reaching points on a plan), and we refer to the corresponding paper for a thorough analysis of these results.

5 Conclusion

We have integrated in this paper two important exploration principles of developmental robotics (exploration in the sensory space and active learning based on an empirical measure of the competence progress) into an integrated probabilistic framework able to express various exploration strategies in a compact and unified manner. This allowed quantitative comparisons of these strategies, showing that an active goal exploration is the most efficient to reach a set of goals uniformly sampled in the reachable part of the sensory space –as already shown in previous works of our team.

Further works should rely the approach to other tentatives of exploration strategy unification (e.g. Lopes and Oudeyer (2012); Oudeyer and Kaplan (2007)). We also want to study the effect of an online adaptation of the choice space, taking advantage of the fact that our formalism does not restrict it to be either M or S. For example, we could study how the agent iteratively adapts which part of the sensorimotor space it is interested in at a given time of its development,

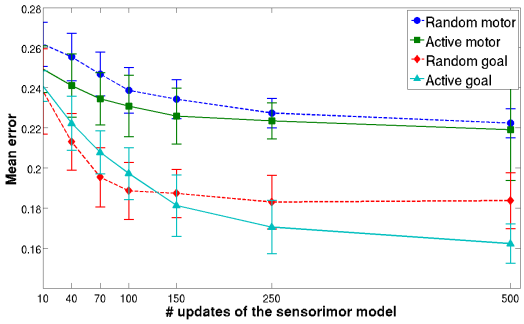


Figure 1: Performance comparison of the four exploration strategies. X-axis: number of update of the sensorimotor model. Y-axis: Mean error on a control task where an agent has to reach 30 test points uniformly distributed in the reachable area of S. For each evaluation point $s_g \in S$, the agent infers 10 motor commands in M from the distribution $p(M \mid s_g\ \omega_{SM})$, where ω_{SM} is the state of the sensorimotor model at the corresponding time step (number of update on the X axis). The error of an agent at a time step is the mean distance between the sensory points actually reached by the 10 motor commands and the evaluation point s_g. Each curve plots the mean and standard deviation of the error for 10 independent simulations with different random seeds, for each of the four exploration strategies described in the previous sections.

favoring exploration in sensorimotor *dimensions* which display higher measures of learning progress. Finally, we are currently extending the implementation to learn how to control sequences of motor commands.

References

Baranes, A. and Oudeyer, P.-Y. (2012). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*.

Barto, A., Singh, S., and Chenatez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. In *Proc. 3rd Int. Conf. Dvp. Learn.*, pages 112–119, San Diego, CA.

Berlyne, D. E. (1954). A theory of human curiosity. *British Journal of Psychology*, 45:180–191.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

Deci, E. and Ryan, R. M. (1985). *Intrinsic Motivation and self-determination in human behavior*. Plenum Press, New York.

Guenther, F. H., Ghosh, S. S., and Tourville, J. A. (2006). Neural modeling and imaging of the cortical interactions underlying syllable production. *Brain and language*, 96(3):280–301.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press.

Lebeltel, O., Bessiere, P., Diard, J., and Mazer, E. (2004). Bayesian robot programming. *Autonomous Robots*, 16:4979.

Lopes, M. and Oudeyer, P.-Y. (2012). The strategic student approach for life-long exploration and learning. In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–8. IEEE.

Moulin-Frier, C. and Oudeyer, P.-Y. (2013a). Exploration strategies in developmental robotics: a unified probabilistic framework. In *International Conference on Development and Learning, Epirob, Osaka, Japan*.

Moulin-Frier, C. and Oudeyer, P.-Y. (2013b). The role of intrinsic motivations in learning sensorimotor vocal mappings: a developmental robotics study. In *Proceedings of Interspeech*, page In press, Lyon, France.

Oudeyer, P.-Y. and Kaplan, F. (2007). What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286.

Rolf, M., Steil, J., and Gienger, M. (2010). Goal babbling permits direct learning of inverse kinematics. *IEEE Trans. Autonomous Mental Development*, 2(3):216–229.

Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In Meyer, J. A. and Wilson, S. W., editors, *Proc. SAB'91*, pages 222–227.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

Thrun, S. (1995). Exploration in active learning. *Handbook of Brain Science and Neural Networks*, pages 381–384.

Around Inverse Reinforcement Learning and Score-based Classification

Matthieu Geist
IMS - MaLIS Research Group (Supélec)
Metz, France
matthieu.geist@supelec.fr

Edouard Klein*
ABC Team (LORIA)
Nancy, France
edouard.klein@supelec.fr

Bilal Piot†
UMI 2958 (GeorgiaTech - CNRS)
Metz, France
bilal.piot@supelec.fr

Yann Guermeur
ABC Team (LORIA)
Nancy, France
yann.guermeur@loria.fr

Olivier Pietquin‡
UMI 2958 (GeorgiaTech - CNRS)
Metz, France
olivier.pietquin@supelec.fr

Abstract

Inverse reinforcement learning (IRL) aims at estimating an unknown reward function optimized by some expert agent from interactions between this expert and the system to be controlled. One of its major application fields is imitation learning, where the goal is to imitate the expert, possibly in situations not encountered before. A classic and simple way to handle this problem is to see it as a classification problem, mapping states to actions. The potential issue with this approach is that classification does not take naturally into account the temporal structure of sequential decision making. Yet, many classification algorithms consist in learning a *score function*, mapping state-action couples to values, such that the value of the action chosen by the expert is higher than the others. The *decision rule* of the classifier maximizes the score over actions for a given state. This is curiously reminiscent of the *state-action value function* in reinforcement learning, and of the associated *greedy policy*.

Based on this simple statement, we propose two IRL algorithms that incorporate the structure of the sequential decision making problem into some classifier in different ways. The first one, SCIRL (Structured Classification for IRL), starts from the observation that linearly parameterizing a reward function by some features imposes a linear parametrization of the Q-function by a so-called feature expectation. SCIRL simply uses (an estimate of) the expert feature expectation as the basis function of the score function. The second algorithm, CSI (Cascaded Supervised IRL), applies a reversed Bellman equation (expressing the reward as a function of the Q-function) to the score function outputted by any score-based classifier, which reduces to a simple (and generic) regression step. These two algorithms come with theoretical guarantees and perform competitively on toy problems.

Keywords: Inverse Reinforcement Learning, Score-based Classification.

Acknowledgements

The research leading to these results has received partial funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°270780.

*Edouard Klein is also with the IMS - MaLIS Research Group
†Bilal Piot is also with the IMS - MaLIS Research Group
‡Olivier Pietquin is also with the IMS - MaLIS Research Group

1 Introduction

Inverse reinforcement learning (IRL) aims at estimating an unknown reward function optimized by some expert agent from interactions between this expert and the system to be controlled. One of its major application fields is imitation learning, where the goal is to imitate the expert, possibly in situations not encountered before. A classic and simple way to handle this problem is to see it as a classification problem, mapping states to actions. The potential issue with this approach is that classification does not take naturally into account the temporal structure of sequential decision making. Yet, many classification algorithms consist in learning a *score function*, mapping state-action couples to values, such that the value of the action chosen by the expert is higher than the others. The *decision rule* of the classifier maximizes the score over actions for a given state. This is curiously reminiscent of the *state-action value function* in reinforcement learning, and of the associated *greedy policy*.

Based on this simple statement, we propose two IRL algorithms that incorporate the structure of the sequential decision making problem into some classifier in different ways. The first one, SCIRL (Structured Classification for IRL), starts from the observation that linearly parameterizing a reward function by some features imposes a linear parametrization of the Q-function by a so-called feature expectation. SCIRL simply uses (an estimate of) the expert feature expectation as the basis function of the score function. The second algorithm, CSI (Cascaded Supervised IRL), applies a reversed Bellman equation (expressing the reward as a function of the Q-function) to the score function outputted by any score-based classifier, which reduces to a simple (and generic) regression step. These two algorithms come with theoretical guarantees and perform competitively on toy problems.

2 From Markov Decision Processes...

A Markov Decision process (MDP) [11] is a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ where $\mathcal{S}$ is the finite state space[1], $\mathcal{A}$ the finite actions space, $\mathcal{P} = \{P_a = (p(s'|s,a))_{1\leq s,s'\leq|\mathcal{S}|}, a \in \mathcal{A}\}$ the set of Markovian transition probabilities, $\mathcal{R} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the state-action reward function and γ the discount factor. A deterministic policy $\pi \in \mathcal{S}^{\mathcal{A}}$ defines the behavior of an agent. The quality of this control is quantified by the value function $v^\pi_\mathcal{R} \in \mathbb{R}^\mathcal{S}$, associating to each state the cumulative discounted reward for starting in this state and following the policy π afterwards: $v^\pi_\mathcal{R}(s) = \mathbb{E}[\sum_{t\geq 0}\gamma^t\mathcal{R}(S_t, A_t)|S_0 = s, \pi]$. An optimal policy $\pi^*_\mathcal{R}$ (according to the reward function $\mathcal{R}$) is a policy of associated value function $v^*_\mathcal{R}$ satisfying $v^*_\mathcal{R} \geq v^\pi_\mathcal{R}$, for any policy π and componentwise.

Let P_π be the stochastic matrix $P_\pi = (p(s'|s,\pi(s)))_{1\leq s,s'\leq|\mathcal{S}|}$ and $\mathcal{R}_\pi$ the reward function defined as $\mathcal{R}_\pi(s) = \mathcal{R}(s,\pi(s))$. With a slight abuse of notation, we may write a the policy which associates the action a to each state s. The Bellman evaluation (resp. optimality) operator $T^\pi_\mathcal{R}$ (resp. $T^*_\mathcal{R}$) : $\mathbb{R}^\mathcal{S} \to \mathbb{R}^\mathcal{S}$ is defined as $T^\pi_\mathcal{R} v = \mathcal{R}_\pi + \gamma P_\pi v$ (resp. $T^*_\mathcal{R} v = \max_\pi T^\pi_\mathcal{R} v$). These operators are contractions and $v^\pi_\mathcal{R}$ and $v^*_\mathcal{R}$ are their respective fixed-points: $v^\pi_\mathcal{R} = T^\pi_\mathcal{R} v^\pi_\mathcal{R}$ and $v^*_\mathcal{R} = T^*_\mathcal{R} v^*_\mathcal{R}$. The action-value function $Q^\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ adds a degree of freedom on the choice of the first action, it is formally defined as $Q^\pi_\mathcal{R}(s,a) = [T^a_\mathcal{R} v^\pi_\mathcal{R}](s)$. Let $Q^*_\mathcal{R}$ be the optimal state-action value function, an important property is that any optimal policy $\pi^*_\mathcal{R}$ is greedy respectively to it:

$$\pi^*_\mathcal{R}(s) \in \operatorname*{argmax}_{a\in\mathcal{A}} Q^*_\mathcal{R}(s,a). \tag{1}$$

Reinforcement learning and approximate dynamic programming aim at estimating the optimal control policy $\pi^*_\mathcal{R}$ when the model (transition probabilities and the reward function) is unknown (but observed through interactions with the system to be controlled) and when the state space is too large to allow exact representations of the objects of interest (as value functions or policies) [2, 12, 14]. We refer to this as the direct problem. On the contrary, (approximate) inverse reinforcement learning [10] aims at estimating a reward function for which an observed policy is (nearly) optimal. Let us call this policy the expert policy, denoted π_E. We may assume that it optimizes some unknown reward function $\mathcal{R}_E$. The aim of IRL is to compute some reward $\hat{\mathcal{R}}$ such that the expert policy is (close to be) optimal, that is such that $v^*_{\hat{\mathcal{R}}} \approx v^{\pi_E}_{\hat{\mathcal{R}}}$. We refer to this as the inverse problem. This is an ill-posed problem, for which many approaches have been proposed (many of them try to learn a reward function such that the related optimal policy matches some measure of the distribution induced by the expert policy, see for example [9] for a brief overview).

3 ... to Classification

A major application of IRL is imitation learning, which aims at generalizing the observed behavior of the expert controller π_E. Using IRL, this could be done by searching for an optimal policy according to the estimated reward $\hat{\mathcal{R}}$. A more classic approach is to cast imitation as a supervised learning problem. Assume that a trajectory $\{(s_i, a_i = \pi_E(s_i), s_{i+1})_{1\leq i\leq N}\}$ drawn by the expert is available. As the action set is finite (and usually small), one can

[1] This work can be extended to compact state spaces, up to some technical aspects.

train a classifier on the dataset $\{(s_i, a_i)_{1 \leq i \leq N}\}$. A classifier learns a decision rule π_c which aims at minimizing the classification error $\epsilon_c = \mathbb{E}_{s \sim \rho_E}[\chi_{\pi_c(s) \neq \pi_E(s)}]$, with χ being the indicator function and ρ_E the stationary distribution of the policy π_E. To do so, many approaches (*e.g.*, multi-class support vector machines [5], structured classification [15], *etc.*) actually learn a score function $q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ from the dataset, ideally satisfying $q(s, \pi(s)) > q(s, a)$, for any state s and any non-optimal action $a \neq \pi_E(s)$. The decision rule is deduced from the learnt score function as

$$\pi_c(s) \in \operatorname*{argmax}_{a \in A} q(s, a). \tag{2}$$

One can notice the similarity between Equations 1 and 2. If seeing the Q-function as a score function (ranking actions according to the expected discounted cumulative rewards) is not new, seing the classifier's score function as a state-action value function (for some unknown reward) is less usual. This can bring some ideas for new IRL algorithms, as exemplified in the next sections.

4 SCIRL (Structured Classification for IRL)

Recall that IRL aims at estimating a reward function. Assume that this reward is linearly parameterized by a set of features ϕ, $\mathcal{R}_\theta(s, a) = \theta^\top \phi(s, a)$. This naturally leads to a parametrization of the state-action value function, for any policy π:

$$\begin{aligned} Q^\pi_{\mathcal{R}_\theta}(s, a) &= \mathbb{E}[\sum_{t \geq 0} \gamma^t \mathcal{R}_\theta(S_t, A_t) | S_0 = s, A_0 = a, \pi] = \theta^\top \mathbb{E}[\sum_{t \geq 0} \gamma^t \phi(S_t, A_t) | S_0 = s, A_0 = a, \pi] \\ &= \theta^\top \mu_\pi(s, a) \\ \text{with } \mu_\pi(s, a) &= \mathbb{E}[\sum_{t \geq 0} \gamma^t \phi(S_t, A_t) | S_0 = s, A_0 = a, \pi] \end{aligned} \tag{3}$$

being called the feature expectation. To sum up, choosing a parametrization for the reward imposes a parametrization for the state-action value function of any policy, and notably for the one of the expert policy, $Q^{\pi_E}_{\mathcal{R}_\theta}(s, a) = \theta^\top \mu_E(s, a)$ (using μ_E as a shorthand for μ_{π_E}). For the unknown reward function $\mathcal{R}_E$, the expert (assumed optimal) policy π_E is greedy respectively to $Q^{\pi_E}_{\mathcal{R}_E}$. Recalling Eq. 2, it is therefore quite natural to parameterize the score function of the classifier with μ_E as linear features. This is the basic principle of the SCIRL algorithm [7], which can be summarized as follows:

1. parameterize the score function with the expert feature expectation: $q_\theta(s, a) = \theta^\top \mu_E(s, a)$;
2. learn the parameter vector θ from the dataset $\{(s_i, a_i = \pi_E(s_i))_{1 \leq i \leq N}\}$, such as minimizing the classification error ϵ_c;
3. output the reward $\mathcal{R}_\theta$ (θ being the learnt parameters).

Obviously, the knowledge of μ_E is not a reasonable assumption. However, one can notice the similarity between the feature expectation (Eq. 3) and a value function. Estimating μ_E essentially reduces to an (on-policy) policy evaluation problem [6], which can be done using standard approaches such as the Least-Squares Temporal Differences (LSTD) algorithm [4].

If the idea underlying SCIRL is quite intuitive, one can wonder if it comes with some sort of guarantees. The answer is positive. One can show that, if the classification error is small and if the feature expectation is well estimated, then the expert policy π_E will be near optimal for the learnt reward $\mathcal{R}_\theta$. More formally, we have

$$0 \leq \mathbb{E}_{s \sim \rho_E}[v^*_{\mathcal{R}_{\theta_c}}(s) - v^{\pi_E}_{\mathcal{R}_{\theta_c}}(s)] \leq \frac{C_f}{1 - \gamma} \left(\epsilon_Q + \epsilon_c \frac{2\gamma \|\mathcal{R}_{\theta_c}\|_\infty}{1 - \gamma} \right),$$

with C_f being a standard concentration coefficient, ϵ_c being the already defined classification error and ϵ_Q quantifying how well the expert feature expectation is estimated. See [7] for details. This algorithm also performs well empirically, see Sec. 6 and [7].

5 CSI (Cascaded Supervised IRL)

The CSI algorithm [8] explores another way to combine classification with the temporal structure of sequential decision making. Assuming that the optimal state-action value function is known (for some unknown reward $\mathcal{R}$), the reward can easily be estimated by reversing the Bellman optimality equation:

$$\mathcal{R}(s, a) = Q^*_{\mathcal{R}}(s, a) - \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) \max_{a' \in \mathcal{A}} Q^*_{\mathcal{R}}(s', a').$$

Assume that using some classifier, a score function q has been learnt (with associated –greedy– decision rule π_c). Then, one can compute a reward function $\mathcal{R}_c$ associated to this score function as follows:

$$\begin{aligned}\mathcal{R}_c(s,a) &= q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} q(s',a') \\ &= q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) q(s', \pi_c(s')).\end{aligned}$$

However, it is usually not reasonable to assume that the dynamics is known. The reward $\mathcal{R}_c$ can still be estimated using any regression algorithm. Assume that a transition set $\{(s_j, a_j, s'_j)_{1 \leq j \leq M}\}$ is available (ideally such that the distribution over (s_j, a_j) is as uniform as possible), then $r_j = q(s_j, a_j) - \gamma q(s'_j, \pi_c(s'_j))$ is an unbiased estimate of $\mathcal{R}_c(s_j, a_j)$ and an estimate $\hat{\mathcal{R}}_c$ can be computed using any regressor from the following training set

$$\left\{ \left((s_j, a_j), r_j = q(s_j, a_j) - \gamma q(s'_j, \pi_c(s'_j)) \right)_{1 \leq j \leq M} \right\}.$$

The CSI approach can be summarized as follows:

1. estimate a score function q using any score-based classifier trained on the dataset $\{(s_i, a_i = \pi_E(s_i))_{1 \leq i \leq N}\}$;
2. estimate the reward $\hat{R}_c$ using any regressor trained on $\{((s_j, a_j), r_j = q(s_j, a_j) - q(s'_j, \pi_c(s'_j)))_{1 \leq j \leq M}\}$.

CSI is derived from a simple idea, based again on the resemblance between a score function and a state-action value function. Compared to SCIRL, it is more flexible, as any classifier (not necessarily based on a linear parametrization) and any regressor can be used. One can again wonder if this comes with some sort of guarantee, the answer is here also positive. If the classification and the regression error are small enough, then the expert policy π_E is near optimal for the estimated reward $\hat{\mathcal{R}}_c$. More formally, we have

$$0 \leq \mathbb{E}_{s \sim \rho_E}[v^*_{\hat{R}_c}(s) - v^{\pi_E}_{\hat{R}_c}(s)] \leq \frac{1}{1-\gamma} \left(\epsilon_R (1 + C_g) + \epsilon_c \frac{2 \|\hat{\mathcal{R}}_c\|_\infty}{1-\gamma} \right),$$

where C_g is another concentration coefficient (satisfying $C_g \leq C_f$) and where ϵ_R quantifies the regression error (notice that it may be easier to control the term ϵ_R than the term ϵ_Q involved in the SCIRL bound). See [8] for more details. This algorithm also performs well empirically, see Sec. 6 and [8].

6 Experiments

We illustrate the proposed approach on a car driving simulator, similar to [13]. The goal is to drive a car on a busy three-lane highway with randomly generated traffic (driving off-road is allowed on both sides). The car can move left and right, accelerate, decelerate and keep a constant speed. The expert optimizes a handcrafted reward $\mathcal{R}_E$ which favours speed, punish off-road, punishes collisions even more and is neutral otherwise.

We compare SCIRL and CSI to the "relative entropy" algorithm of [3] (which shares with SCIRL and CSI the desired property of not requiring to solve repetitively MDPs, contrary to a large part of the state of the art) and to the unstructured classifier (the one which serves as a basis for both CSI and SCIRL). Algorithms are compared according to an oracle criterion, the mean value function (averaged over a uniform distribution $\mathcal{U}$) of the learnt policy relatively to the unknown reward $\mathcal{R}_E$: $\mathbb{E}_{s \sim \mathcal{U}}[v^\pi_{\mathcal{R}_E}(s)]$, with π the policy outputted by one of the considered algorithms (π optimizes the learnt reward for IRL algorithms).

Results are reported on Fig. 6. IRL approaches work much better than the standard classification. We advocate that this is due to the fact that they take into account the temporal structure of the problem. CSI and SCIRL have similar performances (CSI being slightly –but statistically significantly– better than SCIRL), both are better than the state-of-the-art algorithm of [3].

7 Perspectives

Thanks to the similarity between score functions in classification and state-action value functions in reinforcement learning, SCIRL and CSI have been introduced. Compared to the state of the art, they are quite generic (in the sense that instantiations of these approaches can be derived by "plugging" a large class of well-studied supervised learning methods) and they do not require solving repetitively the direct RL problem (which is a common drawback of most of other algorithms). Both approaches come with theoretical guarantees and perform competitively on toy problems.

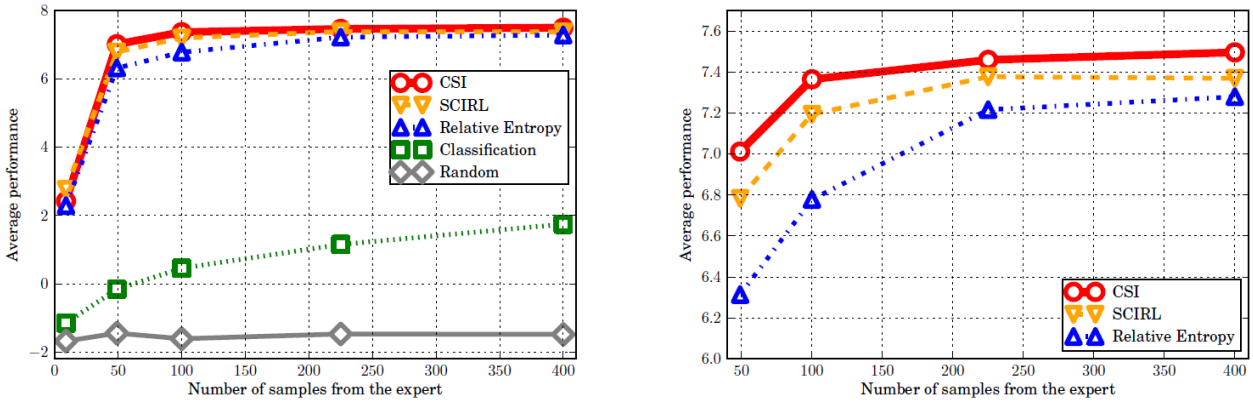


Figure 1: Results on the Highway experiment (the right panel being a zoom of the left one)

.

We plan to apply SCIRL and CSI to more challenging problems, notably to robotics and to ALE (the Arcade Learning Environment [1]). We also plan to study more deeply the theoretical aspects of these algorithms, notably if the bounds we have are tight and how these error propagations can be used to get a more practical finite sample analysis. Another perspective is to propose new algorithms based on the resemblance between score functions and state-action value functions. For example, CSI can be designed with a support vector machine (SVM) for the classification and a support vector regressor (SVR) for the regression. The two related mathematical programs can be "merged", and we are currently studying this alternative IRL algorithm.

References

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *ArXiv e-prints*, 2012.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, 1996.

[3] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *International Conference on. Artificial Intelligence and Statistics (AISTATS)*, 2011.

[4] Steven J. Bradtke and Andrew G. Barto. Linear Least-Squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[5] Yann Guermeur. A generic model of multi-class support vector machine. *International Journal of Intelligent Information and Database Systems (IJIIDS)*, 6(6):555–577, 2012.

[6] Edouard Klein, Matthieu Geist, and Olivier Pietquin. Batch, Off-policy and Model-free Apprenticeship Learning. In *European Workshop on Reinforcement Learning (EWRL)*, 2011.

[7] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse Reinforcement Learning through Structured Classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[8] Edouard Klein, Bilal Piot, Matthieu Geist, and Olivier Pietquin. A cascaded supervised learning approach to inverse reinforcement learning. In *European Conference on Machine Learning (ECML)*, 2013.

[9] Gergely Neu and Czaba Szepesvari. Training Parsers by Inverse Reinforcement Learning. *Machine Learning*, 77(2-3):303–337, 2009.

[10] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2000.

[11] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 3rd edition, March 1998.

[13] Umar Syed and Robert Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[14] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan and Claypool, 2010.

[15] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning Structured Prediction Models: a Large Margin Approach. In *International Conference on Machine Learning (ICML)*, 2005.

Temporal-Difference Learning to Assist Human Decision Making during the Control of an Artificial Limb

Ann L. Edwards
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8
ann.edwards@ualberta.ca

Alexandra Kearney
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8
kearney@ualberta.ca

Michael Rory Dawson
Glenrose Rehabilitation Hospital
Edmonton, AB, Canada, T5B 0B7
mrd1@ualberta.ca

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

Patrick M. Pilarski*
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8
pilarski@ualberta.ca

Abstract

In this work we explore the use of reinforcement learning (RL) to help with human decision making, combining state-of-the-art RL algorithms with an application to prosthetics. Managing human-machine interaction is a problem of considerable scope, and the simplification of human-robot interfaces is especially important in the domains of biomedical technology and rehabilitation medicine. For example, amputees who control artificial limbs are often required to quickly switch between a number of control actions or modes of operation in order to operate their devices. We suggest that by learning to anticipate (predict) a user's behaviour, artificial limbs could take on an active role in a human's control decisions so as to reduce the burden on their users. Recently, we showed that RL in the form of general value functions (GVFs) could be used to accurately detect a user's control intent prior to their explicit control choices. In the present work, we explore the use of temporal-difference learning and GVFs to predict when users will switch their control influence between the different motor functions of a robot arm. Experiments were performed using a multi-function robot arm that was controlled by muscle signals from a user's body (similar to conventional artificial limb control). Our approach was able to acquire and maintain forecasts about a user's switching decisions in real time. It also provides an intuitive and reward-free way for users to correct or reinforce the decisions made by the machine learning system. We expect that when a system is certain enough about its predictions, it can begin to take over switching decisions from the user to streamline control and potentially decrease the time and effort needed to complete tasks. This preliminary study therefore suggests a way to naturally integrate human- and machine-based decision making systems.

Keywords: Temporal-Difference Learning, Human-Machine Interaction, Prediction-based Decision Making, Decision Support, Control Systems, Assistive Rehabilitation Robotics, Nexting

Acknowledgements

The authors gratefully acknowledge support from the Alberta Innovates Centre for Machine Learning, Alberta Innovates – Technology Futures, and the Glenrose Rehabilitation Hospital Foundation. We also thank Thomas Degris and Jason P. Carey for a number of helpful discussions leading up to this work, and Adam Parker for his technical assistance and development on the exArm robotic system.

*Please direct correspondence to Patrick M. Pilarski. All authors are affiliated with the Alberta Innovates Centre for Machine Learning (AICML) and the Reinforcement Learning & Artificial Intelligence Laboratory (RLAI), University of Alberta. Pilarski is also affiliated with the Division of Physical Medicine & Rehabilitation, Faculty of Medicine & Dentistry, University of Alberta.

1 Introduction

In this article we explore the use of reinforcement learning (RL) methods to assist in human decision making during the control of a human-robot interface. We suggest that by acquiring and utilizing knowledge about a user's control-related decisions, control systems and human-machine interfaces could begin to take on an active role in a human decision-making so as to reduce the burden on their users. Knowledge about a user and their robotic system can take the form of learned predictions about the interactions between the human and their device.

Learning and maintaining a wide range of predictive sensorimotor knowledge has been demonstrated in recent work on Nexting (Modayil, White, and Sutton 2012) using learned General Value Functions (GVFs; Sutton et al. 2011). An extension of conventional RL value functions, GVFs represent temporally extended predictions about arbitrary signals of interest. GVFs can be learned in real time using standard RL methods, and have been successfully applied to gather anticipatory knowledge during ongoing human-robot interactions (Pilarski et al. 2012, 2013). As shown in our recent work, combining conventional control methods with GVF-derived predictions can potentially reduce the time and effort needed for users to control a switching-based human-machine interface (Pilarski et al. 2012; Pilarski and Sutton 2012).

Observations from motor control in the human brain also suggest that the ongoing prediction of motor control choices could potentially impact the intuitiveness and functionality of hybrid human-machine decision-making systems. There is a strong relationship between sensorimotor prediction and control in the human brain, and anticipated motor outcomes have been suggested as an important factor in generating and improving control (Flanagan et al. 2003). As described by Flanagan et al. (2003) and Wolpert et al. (2001), predictions are thought to be learned by human subjects before they gain control competency. It is possible that similar mechanisms will prove beneficial for human-robot interaction (Fagg et al. 2004). In particular, leveraging learned knowledge stored in GVFs may be a viable way to support the control-related decisions made by a user with regard to their associated device.

As a motivating example, amputees who control artificial limbs are often required to quickly switch between a number of control actions or modes of operation in order to operate their devices. The increasing complexity of their component human-robot interfaces is in fact one of the major barriers to the use of modern artificial limbs. Artificial limbs commonly use recorded muscle signals (electromyographic recordings, or EMG) to actuate the different joints and motors of a robot system. This approach is termed *myoelectric control*. In more advanced myoelectric systems there are fewer EMG recording sites available on an amputee's body than there are degrees of freedom (DOF) in the prosthesis that the user must control (Williams 2011). One solution to this problem has been the use of EMG signals or mechanical toggles to enable a user to manually switch their control influence between the available joints, movements, grasping patterns, or functions available via the robot arm (Figure 1). While this approach has proved viable for functional use, it is often viewed by users as non-intuitive and unnatural, thereby increasing a user's cognitive effort and the time needed to complete a task. One notable example is the myoelectric interface for some commercial hand and forearm prostheses, which often require a sequence of muscle contractions and manual changes by the user to select a desired gripping or pinching pattern. As such, and despite the potential for restoring lost functions, many patients still reject the use of electromechanical artificial limbs (Williams 2011; Micera et al. 2010).

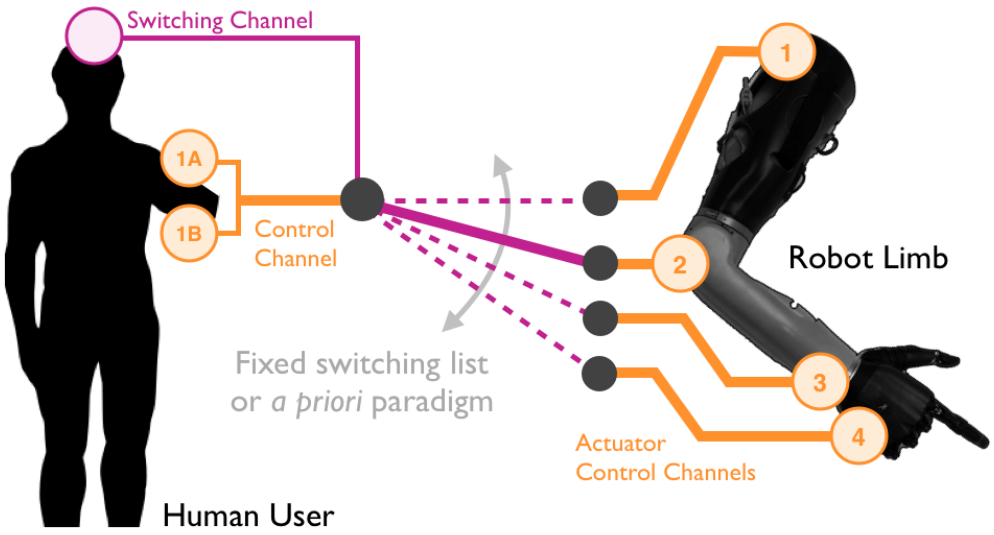


Figure 1: *Example of function switching as used to control an assistive device.* One problem for human-machine interaction occurs when a machine's controllable dimensions outnumber the control channels available to its human user.

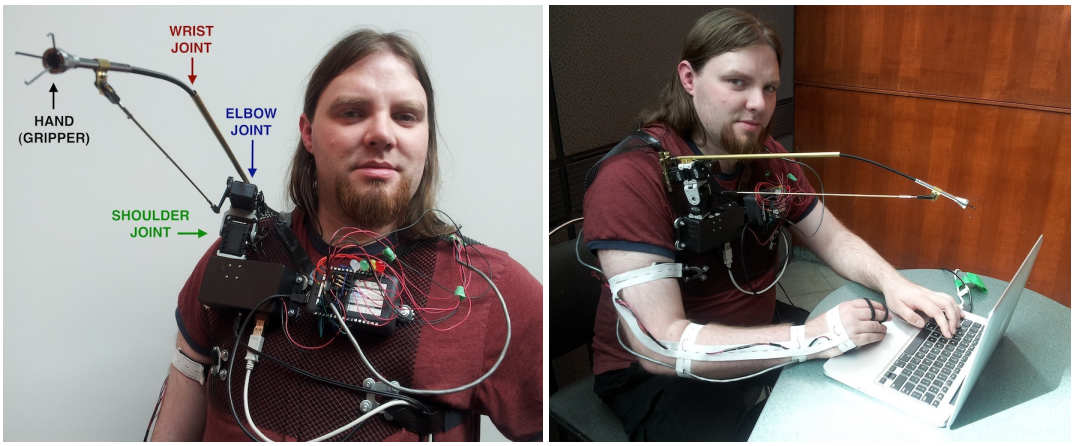


Figure 2: *The experimental platform used in this work*: a wearable robot limb that is controlled using muscle signals from the human body, where the user sequentially controls and switches between the available joints using voluntary muscle contractions (similar to the control interface for a commercial forearm prosthesis).

In the present work, we therefore explore the unification of RL with conventional switching-based control interfaces. In particular, we demonstrate the use of online temporal-difference (TD) learning to predict when a user will switch their control influence between the different control functions of an articulated robot arm. We expect that when a system is certain enough about its predictions regarding a user's switching target and switching timing, it can begin to take over some function switching decisions from the user to streamline control and potentially decrease the time and effort needed to complete tasks. Our over-arching goal is to develop predictive approaches that ultimately enable the more natural control of complex assistive devices.

2 Methods

A wearable, myoelectrically controlled robot arm was used as the experimental platform for this work (Figure 2). This system had four controllable actuators. Two joints of the robot arm could be controlled to move the limb left, right, up, and down, approximating the motions provided by biological shoulder and elbow joints. The lower portion of the arm could also flex inward and outward as in wrist joint movement, and the arm terminated in a simple gripping actuator. Electrodes were affixed to the skin of non-amputee subjects and used to measure EMG signals from four different muscles on the user's body (DE-3.1 double differential electrodes and a Bangoli-8 acquisition system from Delsys, USA). These EMG signals were mapped to two control channels: one to actuate a robotic joint, and one to switch between the different joints in a fixed, sequential fashion.

We examined the ability of GVF-based TD learning to predict joint switching from human interaction with the robot system during simple movement tasks. An able-bodied (non-amputee) subject actuated the myoelectric arm, using electrodes affixed over the wrist flexor and wrist extensor muscles of each arm. Using this wearable system, each subject performed a semi-repetitive motion, moving the robot's shoulder to the right, moving the elbow up and down an arbitrary number of times, moving the shoulder joint back to the left, and then moving the wrist up and down an arbitrary number of times. This H-shaped movement pattern was repeated for 10–30 minutes. As shown in Figure 3, this resulted in a rich stream of data for use by the RL system, and provided a challenging setting for learning due to the temporal variability and non-stationary nature of the user's myoelectric control signals and switching behaviour.

The knowledge learned by our system regarding a user's switching actions took the form of temporally extended predictions about a user's switching prompts; these predictions were similar to the predictions made in our previous work on anticipating the activity of user-controlled actuators (Pilarski et al. 2012; Pilarski and Sutton 2012). Predictions were acquired and updated through multiple offline iterations using an implementation of GVFs (Sutton et al. 2011) and Nexting (Modayil, White, and Sutton 2012). Following the approach of Pilarski et al. (2012), GVFs were updated on each time step using TD learning with eligibility traces and tile-coding function approximation. Each GVF learner was initialized with parameters specifying the prediction of interest, including the timescale of the temporally extended prediction and the target signal of interest—here an on/off signal that was active when the user prompted the system to switch motor functions. Signal sampling and learning updates occurred at 15 Hz (many times per second). The state representation used by the machine learner was comprised of motor feedback (e.g., position, speed) from the robot arm and signals relating to the human's recorded muscle activity (e.g., processed EMG signals and switching cues), as well historical information in the form of decayed traces of these signals. As in previous work, our system also learned a series of temporally extended predictions regarding the motion of each user-driven joint and the user's myoelectric signals (with prediction done in the same way as for switching signal prediction, described above).

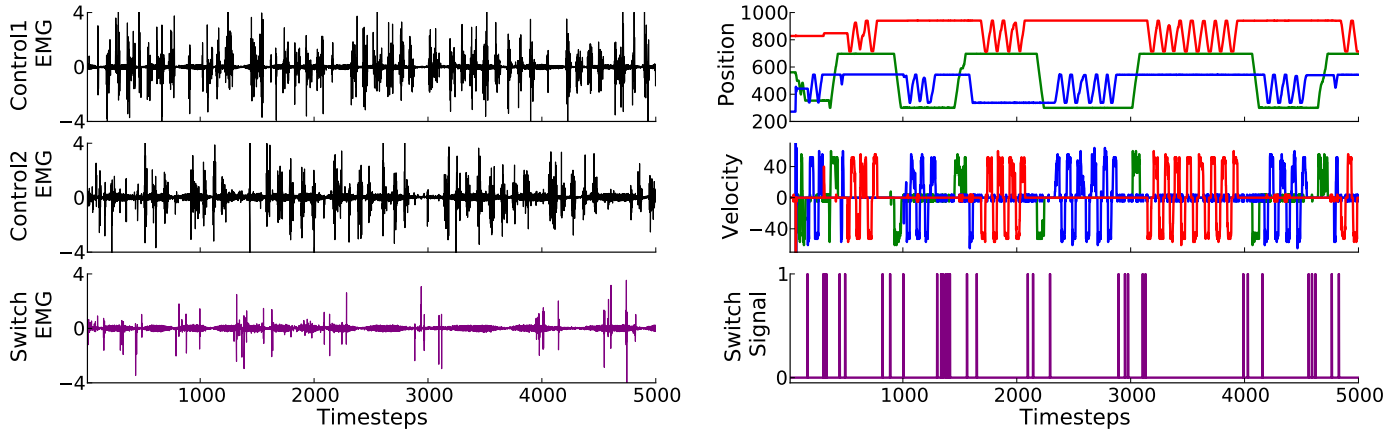


Figure 3: *Example of the sensorimotor data stream from the human-machine interface*, including recorded muscle activity (EMG) for control and switching channels (black and purple traces, left), human switching actions (purple trace, right), and the angle and speed for three of the robot's joints (red, blue, and green traces, right).

3 Results & Discussion

As shown in Figure 4, left, following a period of learning our approach demonstrated the ability to forecast an upcoming switching cue from the human user. Advance knowledge of switching (a rise in the dark purple trace) was observed to arrive a fraction of a second before the actual human-initiated event (grey pulse). Predictions on both training and testing data (dark purple traces) were also observed to begin to approach the true, computed return (light purple trace) as learned progressed. Additional learning is expected to improve the agreement between the true and learned predictions, and testing is ongoing to determine the best state representation for this learning scenario. As expected from previous work, our approach was also able to consistently anticipate which joint a human user would actuate next while performing their task (Figure 4, right). The timescale for all predictions shown in Figure 4 was 10 time steps. Testing and training data were sampled from the same human user, with the training and testing sessions being conducted on different days. The testing data were not seen by the learning system prior to evaluation.

By ranking the magnitude of joint activity predictions prior to manual switching by the user, a learning system is able to determine the most appropriate joint to select at the time of switching (Pilarski et al. 2012). In other words, simple relationships between the predictions can be used to formulate the system's switching suggestions, i.e., which joint to actuate next. These suggestions depend on both context and learned knowledge about a user's preferences. The present work contributes a way to determine the desired timing of switching actions. Taken together, these straightforward applications of learned predictive knowledge provide a way to allow a learning-based control system to gradually assume more autonomy and decision-making responsibility during ongoing human-robot interactions. One useful feature of this approach is that no explicit or time consuming reinforcement is needed from the human user to correct or affirm the learning system's suggested decisions; the use of a mode or function by the user verifies the system's choices, while use of an alternate function decreases the learning system's predictions about the suitability of a given control option (Pilarski and Sutton 2012). Our approach therefore differs from predominant approaches to human-directed RL like human reward (e.g., Thomaz and Breazeal 2008) and demonstration learning (e.g., Lin 1992).

The ability shown in the present work to anticipate a switching event promises to greatly reduce the need to manually initiate switching. The time needed for switching could potentially even be eliminated in certain situations. As suggested in Pilarski and Sutton (2012), removing the need to explicitly switch functions during commonly performed tasks could result in almost as great a time savings as selecting the optimal switching target or function. These expectations remain to be demonstrated in future work with non-amputee and amputee subjects.

4 Conclusions

In this work we demonstrated the use of reinforcement learning (RL) to help with human decision making, and specifically provided a first step towards intuitive human interaction with a switching-based biomedical robot. Function switching is a common way to deal with increasing device complexity, but it poses additional challenges to the natural and efficient control devices by a user. To help address the barriers to streamlined human-robot interactions, we deployed state-of-the-art RL techniques to acquire and maintain knowledge about a user and their robotic system. Our approach was able to build up and maintain forecasts about a user's switching behaviour in real time. We also confirmed previous observations that our approach can detect a user's control intent prior to their explicit control actions. Bringing together

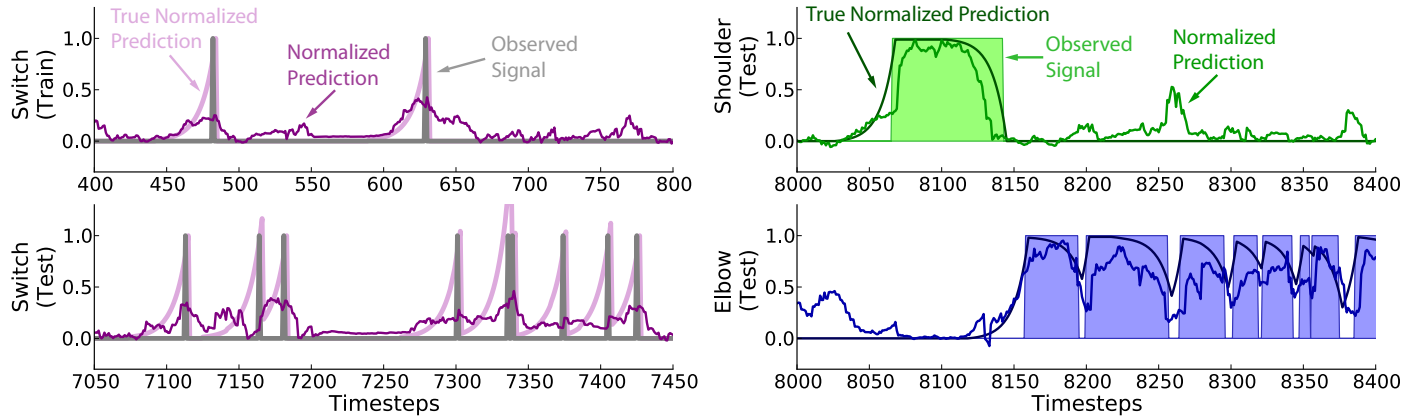


Figure 4: *Example of switching event and joint activity predictions on previously unseen testing data* after five iterations of TD learning through 28k steps of training data. *Left*: switching event predictions begin to rise in advance of actual switching events initiated by the user, as shown on both training and testing data. *Right*: predictions about joint activity rise in advance of expected joint actuation. Modulating a control interface based on these predictions promises to reduce the time needed for a user to complete a task with a multi-joint robot arm (Pilarski et al., 2012, Pilarski and Sutton 2012).

these two ideas, a system could potentially determine what function a user intends to deploy, and when they wish to begin using the new function. Our approach allows a user to remain in direct control of a system while still allowing the device to suggest or initiate increasingly appropriate control options. Furthermore, the opportunity for ongoing yet optional human interaction in the decision-making process provides an intuitive and reward-free way for users to correct or reinforce the decisions made by a semi-autonomous machine learning system. This preliminary study therefore opens the way for naturally blending the control decisions made a human and their assistive robot or other human-machine interface. Future work will continue to pursue the integration of biological and synthetic reinforcement learning and decision-making systems.

References

Fagg, A. H., Rosenstein, M. T., Platt, Jr., R., Grupen, R. A. (2004). Extracting user intent in mixed initiative teleoperator control. In *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, Chicago, Illinois, 2004-6309.

Flanagan, J. R., Vetter, P., Johansson, R. S., Wolpert, D. M. (2003). Prediction precedes control in motor learning. *Current Biology* 13(2): 146–150.

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

Micera, S., Carpaneto, J., Raspopovic, S. (2010). Control of hand prostheses using peripheral information. *IEEE Reviews in Biomedical Engineering* 3: 48–68.

Modayil, J., White, A., Sutton, R. S. (2012). Multi-timescale nexting in a reinforcement learning robot. In *Proceedings of the 2012 Conference on Simulation of Adaptive Behavior*, Odense, Denmark, 299–309.

Pilarski, P. M., Dawson, M. R., Degris, T., Carey, J. P., Sutton, R. S. (2012). Dynamic switching and real-time machine learning for improved human control of assistive biomedical robots. In *Proceedings of the 4th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, June 24–27, Roma, Italy, 296–302.

Pilarski, P. M., Sutton, R. S. (2012). Between instruction and reward: Human-prompted switching. *AAAI 2012 Fall Symposium on Robots Learning Interactively from Human Teachers (RLIHT)*, Nov. 2-4, Arlington, VA, USA, AAAI Technical Report FS-12-07, 46–52.

Pilarski, P. M., Dawson, M. R., Degris, T., Carey, J. P., Chan, K. M., Hebert, J. S., Sutton, R. S. (2013). Adaptive artificial limbs: A real-time approach to prediction and anticipation. *IEEE Robotics & Automation Magazine* 20(1): 53–64.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., Precup, D. (2011). Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2–6, Taipei, Taiwan, 761–768.

Thomaz, A. L., Breazeal, C. (2008). Teachable robots: understanding human teaching behaviour to build more effective robot learners. *Artificial Intelligence* 172: 716–737.

Williams, T. W. (2011). Guest editorial: progress on stabilizing and controlling powered upper-limb prostheses. *Journal of Rehabilitation Research and Development* 48(6): ix–xix.

Wolpert, D. M., Ghahramani, Z., Flanagan, J. R. (2001). Perspectives and problems in motor learning. *Trends Cogn. Sci.* 5(11): 487–494.

Efficient Learning of Mixed Observable Predictive State Representation*

Sylvie Ong,
School of Computer Science
McGill University
Montreal, QC, Canada
song@gmail.com

Yuri Grinberg
School of Computer Science
McGill University
Montreal, QC, Canada
ygrinb@cs.mcgill.ca

Joelle Pineau
School of Computer Science
McGill University
Montreal, QC, Canada
jpineau@cs.mcgill.ca

Abstract

A key to successful reinforcement learning and planning in partially observable domains is a well built representation with a succinct state information. Recently some progress has been made on learning such representations within the framework of PSRs [6], specifically applying the spectral learning approach [3]. These algorithms guarantee to learn a nearly exact model given enough data, while at the same time keeping the state representation size within some well defined bounds. Nevertheless, in many realistic domains these bounds could be prohibitive from the point of view of RL algorithms, requiring domain specific knowledge and problem structure to make further reduction in the size of the state space.

In this work we consider a specific problem structure, termed mixed observability [7]. As opposed to partial observability [5], some of the observed variables are assumed to be Markovian, resulting in a more compact state representation. Mixed observability setting was found useful in domains as diverse as robotics [7], computational sustainability [4] and operations research [8]. Motivated by its broad applicability, in this work we develop a PSR–based spectral learning algorithm that leverages this structural assumption. Beyond providing a more compact state representation, the proposed algorithm is faster and more data efficient as compared to the existing spectral learning methods for PSRs. These advantages are supported by theoretical as well as experimental results.

Keywords: Predictive Representations, Spectral Learning, Reinforcement Learning

Acknowledgements

The authors thank Doina Precup for helpful discussions on this work. Financial support for this research was provided by the NSERC Discovery grant program.

*A full version of this work appears at [8].

1 Partially Observable Domains

Much of the reinforcement learning (RL) literature are devoted to so called *fully–observable* environments. These environments are typically modeled by Markov decision processes (MDP) [9], a formalism that rests on an assumption that the observation received by an agent at the current time step is sufficient to predict its future dynamics. In other words, the agent does not need to keep any information about its past observations.
Despite the ubiquity of domains where the MDP formalism is applicable, in practice, many problems are only partially observable. This can be due to insufficient sensing (e.g. noisy sensors in robotics), or the nature of the domain itself (e.g. weather prediction). As such, much of the effort in solving RL problems goes into building a proper model of the environment that provides some notion of state.
A common framework for partially observable domains with discrete actions and observations is the *partially observable Markov decision process* (POMDP) [5]. In this framework the environment dynamics are still modelled by an MDP, but the actual state is hidden and instead the agent observes only some, possibly stochastic, information about the state of the system. As a result, the state representation in POMDPs at any point of time is a distribution over hidden states, also called *belief state*. While the problem of planning in POMDPs has been addressed throughout the years (e.g. [5], [10]), learning POMDP representations directly from data has always been a difficult task. The classical approach was based on the expectation maximization method [2], which is a locally optimal method and as such initialization–dependent.

1.1 Predictive State Representations

Predictive state representations (PSR) were introduced as a means to represent a partially observable environment without explicit notion of latent states, with the goal of developing efficient learning algorithms [6, 11]. Essentially, a predictive representation is only required to keep some form of sufficient statistic from the past to be able to predict the future sequences of observations given sequences of actions. In this paper we consider what is called *linear* PSRs as they can represent environments modelled by POMDPs on one hand, and on the other hand there are efficient algorithms for learning these representations.
Let $\mathcal{A}$ and $\mathcal{O}$ be discrete action and observation spaces correspondingly. Given a sequence of actions $a_1, ..., a_k \in \mathcal{A}$, the environment outputs a sequence of observations $o_1, ..., o_k \in \mathcal{O}$, with probability $\mathrm{P}(o_1, ..., o_k | a_1, ..., a_k)$. The set of parameters

$$\{\mathbf{m}_* \in \mathbb{R}^n, \{\mathbf{M}_{ao} \in \mathbb{R}^{n \times n}\}_{a \in \mathcal{A}, o \in \mathcal{O}}, \mathbf{p}_0 \in \mathbb{R}^n\}$$

defines a n-dimensional linear PSR that represents this environment if the following holds:

$$\forall k \in \mathbb{N}, o_i \in \mathcal{O}, a_j \in \mathcal{A} : \mathrm{P}(o_1, ..., o_k | a_1, ..., a_k) = \mathbf{m}_*^\top \mathbf{M}_{a_k o_k} \cdots \mathbf{M}_{a_1 o_1} \mathbf{p}_0,$$

where $\mathbf{p}_0$ gives the initial state of the PSR [12].
Let $\mathbf{p}(h)$ be the PSR state corresponding to a history of action–observation pairs h. Then, for any $ao \in \mathcal{A} \times \mathcal{O}$, the following recursive state update equation can be derived from the above :

$$\mathbf{p}(hao) \triangleq \frac{\mathbf{M}_{ao} \mathbf{p}(h)}{\mathbf{m}_*^\top \mathbf{M}_{ao} \mathbf{p}(h)}.$$

In this sense, the vector $\mathbf{p}(h)$ represents the state of the system since there is no need to keep the history in memory, specifically:

$$\forall k \in \mathbb{N}, o_i \in \mathcal{O}, a_j \in \mathcal{A} : \mathrm{P}(o_1, ..., o_k | h, a_1, ..., a_k) = \mathbf{m}_*^\top \mathbf{M}_{a_k o_k} \cdots \mathbf{M}_{a_1 o_1} \mathbf{p}(h).$$

Finally, it was shown that the dimension of the linear PSR is at most the same as the number of hidden states in a POMDP representing the same environment [11]. From this point of view, a linear PSR representation can be more compact than a POMDP representation.

1.2 Spectral Learning for Linear PSRs

Most of the learning algorithms for linear PSRs are based on estimating a portion of the *system dynamics matrix* (SDM) [11], a matrix that contains all the information about the observable behavior of the environment. In this matrix the columns correspond to histories and rows correspond to future action–observation sequences (tests), so that the entries are probabilities of observations given the history and a corresponding sequence of actions. Then, the spectral algorithm proceeds with performing a *singular value decomposition* (SVD) to the corresponding portion of the matrix and computes the PSR parameters based on the results of SVD [3]. Let $\mathbf{P}_{\mathcal{T},\mathcal{H}} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ be an estimated submatrix of SDM for the corresponding set of tests $\mathcal{T}$ and set of histories $\mathcal{H}$ provided with its SVD decomposition; similarly, $\mathbf{P}_{\mathcal{T},ao,\mathcal{H}}$ be a collection of estimated SDM submatrices with tests from $ao\mathcal{T}$ for all $ao \in \mathcal{A} \times \mathcal{O}$; and $\mathbf{p}_\mathcal{H}$ be a vector of probabilities of histories from $\mathcal{H}$. Then, the algorithm computes a PSR representation from the following:

- $\mathbf{p}_0 = \mathbf{S}\mathbf{V}^\top \mathbf{1}$,

- $\mathbf{m}_*^\top = \mathbf{p}_\mathcal{H}^\top (\mathbf{S}\mathbf{V}^\top)^\dagger$,
- $\forall ao \in \mathcal{A} \times \mathcal{O} : \mathbf{M}_{ao} = \mathbf{U}^\top \mathbf{P}_{\mathcal{T},ao,\mathcal{H}} (\mathbf{S}\mathbf{V}^\top)^\dagger$,

where † represents a Moore–Penrose pseudoinverse of a matrix.
As long as $\mathcal{T}$ and $\mathcal{H}$ are sufficiently large sets and $\mathbf{P}_{\mathcal{T},\mathcal{H}}, \mathbf{P}_{\mathcal{T},ao,\mathcal{H}}, \mathbf{p}_\mathcal{H}$ are well estimated from data, the algorithm is guaranteed to produce an exact (up to numerical errors) linear PSR representation of the environment [3].

2 Mixed Observable Predictive State Representations

Despite linear PSRs being potentially more compact than POMDPs, in more realistic problems the state dimension is often still too large to apply planning algorithms. In many problems, one can notice that the partial observability setting may be too general. One promising direction with this respect is the appearance of a mixed observability setting, which was proposed as a structural assumption that simplifies planning by reducing the state space of the problem [7, 1]. Mixed observability can be seen as a middle ground between full observability (MDP) and partial observability, by assuming that only some of the components of an observation satisfy the Markov assumption. Specifically, let the observation space $\mathcal{O} \triangleq \mathcal{X} \times \mathcal{Z}$ be a product of a fully observable component $\mathcal{X}$ and a partially observable component $\mathcal{Z}$, such that for any history h:

$$\forall a \in \mathcal{A}, xz \in \mathcal{X} \times \mathcal{Z} : \mathrm{P}(xz|h,a) = P(x|\mathbf{p}_z(h), x(h)) \cdot P(z|\mathbf{p}_z(h), x(h), x), \tag{1}$$

where $\mathbf{p}_z(h)$ is the sufficient statistic for the component $\mathcal{Z}$ after observing h (z-th state), and $x(h)$ is the last fully observable component in h. This structure occurs in POMDPs whose state space can be factored in two sets $\mathcal{X}$ and $\mathcal{Y}$, such that the states in $\mathcal{Y}$ produce partial observations $\mathcal{Z}$ while those in $\mathcal{X}$ are fully observable [7]. In this case, $\mathbf{p}_z(h)$ is simply a belief vector over states in $\mathcal{Y}$.
In the above formulation, the sufficient statistic for $\mathcal{Z}$ together with the last fully observable $\mathcal{X}$ represent the state of the environment, as opposed to keeping a sufficient statistic for $\mathcal{X} \times \mathcal{Z}$ together in the general partially observable setting.
From Eq. (1) it is clear that the z-th state dynamics depend on the current x, so in what follows the equations updating z-th state will be different for each $x \in \mathcal{X}$, as expected . The *mixed observable PSR* (MOPSR) is therefore defined by

$$\left\{ \mathbf{x}_0 \in \mathbb{R}^{|\mathcal{X}|}, \left\{ \mathbf{m}_*^q \in \mathbb{R}^{k_q}, \{\mathbf{M}_{axz}^q \in \mathbb{R}^{k_x \times k_q}\}_{a \in \mathcal{A}, xz \in \mathcal{X} \times \mathcal{Z}}, \mathbf{p}_0^q \in \mathbb{R}^{k_q} \right\}_{\forall q \in \mathcal{X}} \right\},$$

if the following holds:

1. $\forall x \in \mathcal{X} : \sum_{z \in \mathcal{Z}} P(xz) = \mathbf{x}_0[x]$, (i.e., distribution over the initial state in $\mathcal{X}$)
2. $\forall k \in \mathbb{N}, x_i z_i \in \mathcal{X} \times \mathcal{Z}, a_j \in \mathcal{A}$:

$$\mathrm{P}(x_1 z_1, ..., x_k z_k | a_1, ..., a_k) = \sum_{q \in \mathcal{X}} \mathbf{x}_0[q] \cdot \left(\mathbf{m}_*^{x_k \top} \mathbf{M}_{a_k x_k z_k}^{x_{k-1}} \cdots \mathbf{M}_{a_2 x_2 z_2}^{x_1} \cdot \mathbf{M}_{a_1 x_1 z_1}^q \mathbf{p}_0^q \right).$$

Since the initial x is unknown, the starting state of the MOPSR is a weighted collection of vectors (note that these can be of different sizes) with coefficients defined by $\mathbf{x}_0$. Hence, for the first action–observation pair axz the state update equation would be

$$\mathbf{p}(axz) = \frac{\sum_{q \in \mathcal{X}} \mathbf{x}_0[q] \cdot \mathbf{M}_{axz}^q \mathbf{p}_0^q}{\mathbf{m}_*^{x \top} \left(\sum_{q \in \mathcal{X}} \mathbf{x}_0[q] \cdot \mathbf{M}_{axz}^q \mathbf{p}_0^q \right)},$$

and for the remaining, one gets the recursive state update equation similar to the usual linear PSR setting:

$$\mathbf{p}(haxz) \triangleq \frac{\mathbf{M}_{axz}^{x(h)} \mathbf{p}(h)}{\mathbf{m}_*^{x \top} \mathbf{M}_{axz}^{x(h)} \mathbf{p}(h)}.$$

The following theorem summarizes the upper bound on the MOPSR representation dimensions.

Theorem 1. *Assume that a POMDP with $|\mathcal{X}| \times |\mathcal{Y}|$ states satisfies the mixed observability assumption* (1)*. Then in general, the dimension of a linear PSR representing this POMDP can be equal to the number of states, while the dimension of each $\mathbf{p}_{q \in \mathcal{X}}$ in a MOPSR representation is upper bounded by $|\mathcal{Y}|$.*

2.1 Spectral Learning for MOPSRs

As we previously mentioned, the last fully observable component influences the future dynamics of the environment in its own way. Not surprisingly, one reasonable and provably correct way to learn MOPSR representations is to first split the estimated submatrices of SDM column–wise into $|\mathcal{X}|$ submatrices, where each of them contains only histories

(columns) that end with a particular fully observable component. Then, SVD is applied to each of these submatrices and the results are used to estimate the parameters of the MOPSR.

For all $q \in \mathcal{X}$, let $\mathbf{P}_{\mathcal{T},\mathcal{H}_q} = \mathbf{U}_q\mathbf{S}_q{\mathbf{V}_q}^\top$ be a SVD decomposition of estimated submatrices of SDM for the set of tests $\mathcal{T}$ and set of histories $\mathcal{H}_q$ that terminate with q; similarly, $\mathbf{P}_{\mathcal{T},axz,\mathcal{H}_q}$ be a collection of estimated SDM submatrices with tests from $axz\mathcal{T}$ for all $axz \in \mathcal{A}\times\mathcal{X}\times Z$; $\mathbf{p}_{\mathcal{H}_q}$ be an estimated vector of probabilities of histories from $\mathcal{H}_q$; and $\mathbf{x}_0$ an estimated distribution over $\mathcal{X}$. Then, the parameters of an MOPSR are obtained using the following ($\forall q \in \mathcal{X}$):

$$\mathbf{p}_0^q = \frac{1}{\mathbf{x}_0[q]}\mathbf{S}_q{\mathbf{V}_q}^\top\mathbf{1}$$
$$\mathbf{m}_*^{q\top} = \mathbf{p}_{\mathcal{H}_q}^\top(\mathbf{S}_q{\mathbf{V}_q}^\top)^\dagger$$
$$\forall axz \in \mathcal{A}\times\mathcal{X}\times\mathcal{Z} : \mathbf{M}_{axz}^q = \mathbf{U}_x^\top\mathbf{P}_{\mathcal{T},axz,\mathcal{H}_q}(\mathbf{S}_q{\mathbf{V}_q}^\top)^\dagger.$$

As with the general spectral learning algorithm, with enough data and diverse enough sets $\mathcal{T}$ and $\mathcal{H}$, one recovers an exact (up to numerical errors) MOPSR representation of the environment. Then, the planning problem can likely be solved using an algorithm similar to one developed for mixed observable setting in POMDPs [7].

From a computational point of view, performing $|\mathcal{X}|$ SVD-s of small matrices is faster than doing SVD on one concatenated matrix due to the nearly cubic complexity of the SVD operation. As a result, using the MOPSR learning outlined above one gets a reduction in complexity by a factor of $|\mathcal{X}|$, if small matrices are roughly equal in size.

Despite a seemingly larger number of parameters to estimate at first glance, the dimensions of matrices and vectors are significantly smaller compared to the linear PSR representation of the same environment, for the same reasons as mentioned above. Overall, a reduction by a factor of $|\mathcal{X}|$ is generally expected here as well. Hence, the proposed algorithm is also expected to be more data efficient, which is evident from the experimental results.

3 Experimental Results

We consider an elevator control problem, which originally appeared at International Probabilistic Planning Competition (IPCC) 2010. It consists of one (or more) elevator(s) operating in a building with people possibly waiting at any floor. All information except for the direction in which people desire to go is fully observable (until the elevator's arrival). We applied both a general spectral learning algorithm (TPSR) and the MOPSR learning algorithm on the problem formulation with one elevator and four floors. Note that although this instance of the problem might be easy for planning, it is nontrivial for learning, since the number of observations is 256 ($|\mathcal{X}| = 64, |\mathcal{Y}| = 4$), 5 actions and total number of states is 64^2, requiring a lot of data to estimate SDM submatrices with longer histories/tests.

While the upper bound on the rank of each $\mathbf{P}_{\mathcal{T},\mathcal{H}_z}$ is 64, with 10,000 training trajectories there were only 48 non-zero

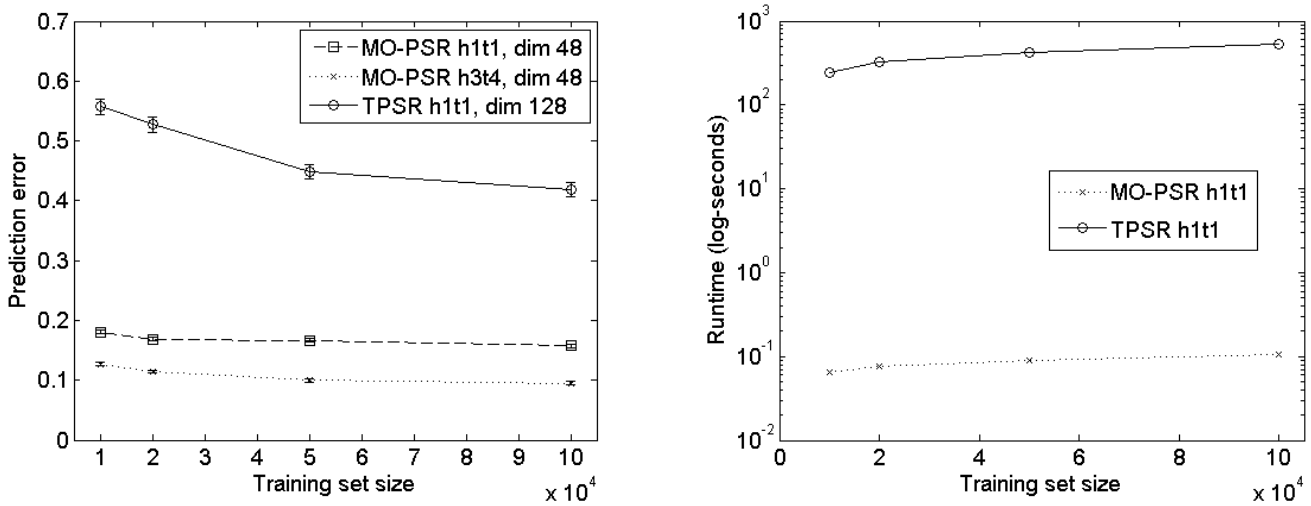


Figure 1: (*a*) Prediction errors of the MOPSR and TPSR models. (*b*) Run times for SVD operation for the MOPSR and TPSR models.

singular values from the SVD on some matrices, so we learned MOPSR models of dimension 48 throughout. The choices for sets $\mathcal{T}$ and $\mathcal{H}$ were essentially driven by observed trajectories with predefined maximum length. We considered all tests and histories of length 1 (h1t1), as well as tests of length 3 and histories of length 4 (h3t4). However, with TPSR learning we could only consider the first option with the cap on the dimension being 128 without running out of memory

(7GB) during SVD operation, while the upper bound on the rank of $\mathbf{P}_{\mathcal{T},\mathcal{H}}$ was 64^2.
As shown in Figure 1 (a), the MOPSR model learned from the same amount of data and length 1 histories and tests outperformed the TPSR model in terms of the prediction errors. The MOPSR model based on longer histories and test performed even better as expected. Furthermore, the MOPSR learning was much faster compared to TPSR learning, as can be seen from Figure 1 (b), due to performing SVD on large matrices in the TPSR algorithm.

See [8] for a detailed version of this paper including the proofs and more experimental results.

References

[1] Mauricio Araya-López, Vincent Thomas, Olivier Buffet, and François Charpillet, *A closer look at momdps*, Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on, vol. 2, IEEE, 2010, pp. 197–204.

[2] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss, *A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains*, The annals of mathematical statistics **41** (1970), no. 1, 164–171.

[3] B. Boots, S. Siddiqi, and G. Gordon, *Closing the learning-planning loop with predictive state representations*, Proceedings of Robotics: Science and Systems, 2010.

[4] Iadine Chades, Josie Carwardine, Tara G Martin, Samuel Nicol, Régis Sabbadin, Olivier Buffet, et al., *Momdps: a solution for modelling adaptive management problems*, AAAI, 2012.

[5] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra, *Planning and acting in partially observable stochastic domains*, Artificial Intelligence **101** (1998), 99–134.

[6] Michael Littman, Richard Sutton, and Satinder Singh, *Predictive representations of state*, Advances in Neural Information Processing Systems (NIPS), 2002.

[7] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, *Planning under uncertainty for robotic tasks with mixed observability*, International Journal of Robotics Research (2010).

[8] Sylvie Ong, Yuri Grinberg, and Joelle Pineau, *Mixed observability predictive state representations*, AAAI, 2013.

[9] Martin L Puterman, *Markov decision processes*, Handbooks in Operations Research and Management Science **2** (1990), 331–434.

[10] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, *Online planning algorithms for POMDPs*, JAIR **32** (2008), 663–704.

[11] S. Singh, M. James, and M. Rudary, *Predictive state representations: A new theory for modeling dynamical systems*, Proceedings UAI, 2004.

[12] E.W. Wiewiora, *Modeling probability distributions with predictive state representations*, Ph.D. thesis, The University of California at San Diego, 2007.

Approximate Policy Iteration with Demonstration Data

Beomjoon Kim
School of Computer Science
McGill University
Montreal, Quebec, Canada

Amir-massoud Farahmand
School of Computer Science
McGill University
Montreal, Quebec, Canada

Joelle Pineau
School of Computer Science
McGill University
Montreal, Quebec, Canada

Doina Precup
School of Computer Science
McGill University
Montreal, Quebec, Canada

Abstract

We propose an algorithm to solve uncertain sequential decision-making problems that utilizes two different types of data sources. The first is the data available in the conventional reinforcement learning setup: an agent interacts with the environment and receives a sequence of state transition samples alongside the corresponding reward signal. The second data source, which differentiates the setup of this work from the usual reinforcement learning framework, is in the form of expert's demonstrations, that is, a set of states with the expert's suggested actions.

Benefitting from both sources of data, which are available in many real-world application domains, allows the agent to perform well even with few data points. The algorithm is couched in the framework of Approximate Policy Iteration. Its approximate policy evaluation step is formulated as a convex optimization problem in which the expert demonstration data act as a set of linear constraints. In a real robotic navigation task, we show that the algorithm outperforms both pure approximate policy iteration and supervised learning.

Keywords: Reinforcement Learning, Learning from Demonstration, Approximate Policy Iteration, Large-Margin Algorithm, Regularization, Robotics

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the Discovery program, the NSERC Canadian Field Robotics Network (NCFRN), and the NSERC Postdoctoral Fellowship as well as the Canadian Institutes for Health Research (CIHR) through the CanWheel team.

1 Introduction

Solving uncertain sequential decision-making and reinforcement learning (RL) [1] problems with large state spaces can be quite difficult. These are, however, the problems that appear most often in real-world applications, e.g., in robotics, designing treatment strategy for chronic patients, etc. There are two key insights that help us solve these problems. The first is that the algorithm has to benefit from the intrinsic regularities of the problem in hand, and preferably does this adaptively. Even though this idea has been known for a long time in statistics and supervised machine learning, researchers have only recently started to develop such algorithms for RL problems, e.g., Farahmand et al. [2], Taylor and Parr [3], Kolter and Ng [4], Ghavamzadeh et al. [5], Farahmand and Szepesvári [6]. This paper introduces another idea: For some problems, we might occasionally be able to provide the agent with some extra information to guide its learning. In particular, we might provide the agent with the information about what actions are good or close to optimal in a few states.

This extra information, which we call "expert data", is common in many application domains and not using it limits the range of sequential decision-making problems that can be solved. As an example in robot learning, it is a common practice to solicit suggestions from an expert to learn complex behaviour, as done in the Learning from Demonstration (LfD) framework. In robotics and other complex control problems it is important to achieve good performance from relatively little data. It is also particularly crucial to limit the risk involved in learning by trial-and-error (as is done in RL), which could lead to catastrophic failures. A combination of trial-and-error RL data and expert data (i.e., mixing RL and LfD) offers a tantalizing way to effectively address challenging real-world policy learning problems.

Our primary contribution is a new large-margin algorithm that allows us to benefit from the demonstration data in an Approximate Policy Iteration (API) framework. The method is formulated as a coupled convex optimization. The key insight is that one can incorporate the expert demonstration data as a set of linear constraints. The optimization is formulated in a way that permits mistakes in the data provided by the expert, and also accommodates variable availability of expert data (i.e., just an initial batch or continued demonstrations). The algorithm has a theoretical guarantee in the form of an upper bound on the Bellman error, but we do not report that result in this extended abstract (cf. Kim et al. [7]).

We evaluate the algorithm's practicality in a real robot path finding task, where there are a few demonstrations, and trial-and-error data is expensive due to limited time. In all of the experiments, our method performed better than Least-Square Policy Iteration (LSPI) [8], using fewer trial-and-error data points and exhibiting significantly less variance. More empirical studies are reported in [7].

2 APID Algorithm

We consider a *continuous-state, finite-action discounted MDP* $(\mathcal{X}, \mathcal{A}, P, \mathcal{R}, \gamma)$, where $\mathcal{X}$ is a measurable state space (e.g., a subset of $\mathbb{R}^d$), $\mathcal{A}$ is a finite set of actions, $P : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$ is the transition model, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathbb{R})$ is the reward model, and $\gamma \in [0, 1)$ is a discount factor.[1] Let $r(x, a) = \mathbb{E}[\mathcal{R}(\cdot|x, a)]$, and assume that r is uniformly bounded by $R_{\max}$. A measurable mapping $\pi : \mathcal{X} \to \mathcal{A}$ is called a (deterministic) *policy*. As usual, V^π and Q^π denote the value and action-value function for π, while V^* and Q^* denote the corresponding value functions for the optimal policy π^*.

Our algorithm is couched in the framework of Approximate Policy Iteration (API) [9]. A standard API algorithm starts with an initial policy π_0. At the $(k+1)^{\text{th}}$ iteration, given a policy π_k, the algorithm approximately evaluates π_k to find $\hat{Q}_k$, usually as an approximate fixed point of the Bellman operator T^{π_k}: $\hat{Q}_k \approx T^{\pi_k}\hat{Q}_k$.[2] This is called the approximate *policy evaluation* step. Then, a new policy π_{k+1} is computed, which is greedy with respect to $\hat{Q}_k$. There are several variants of API that mostly differ on how the approximate policy evaluation is performed, a challenging problem for continuous state spaces. Most methods attempt to exploit structure in the value function [2, 3, 5], but in some problems one might have extra information about the structure of good or optimal policies as well, which would ideally be incorporated in the algorithm. This is precisely our case, since we have expert demonstrations.

To develop the algorithm, we start with regularized Bellman error minimization, which is a common flavour of policy evaluation used in API. Suppose that we want to evaluate policy π and we are given a batch $\mathcal{D}_{\text{RL}} = \{(X_i, A_i)\}_{i=1}^n$ containing n examples, and that we know the exact Bellman operator T^π. Then, the new value function $\hat{Q}$ is computed as:

$$\hat{Q} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}^{|\mathcal{A}|}} \|Q - T^\pi Q\|_n^2 + \lambda_Q J^2(Q) \tag{1}$$

where $\mathcal{F}^{|\mathcal{A}|}$ is a set of possible action-value functions, the first term is the squared Bellman error evaluated on data,[3] $J^2(Q)$ is a regularization penalty, which can prevent overfitting when $\mathcal{F}^{|\mathcal{A}|}$ is complex, and $\lambda_Q > 0$ is the regularization

[1] For a space Ω with σ-algebra σ_Ω, $\mathcal{M}(\Omega)$ denotes the set of all probability measures over σ_Ω.

[2] For discrete state spaces, $(T^{\pi_k}Q)(x, a) = r(x, a) + \gamma \sum_{x'} P(x'|x, a)Q(x', \pi_k(x'))$.

[3] $\|Q - T^\pi Q\|_n^2 \triangleq \frac{1}{n}\sum_{i=1}^n |Q(X_i, A_i) - (T^\pi Q)(X_i, A_i)|^2$ with (X_i, A_i) from $\mathcal{D}_{\text{RL}}$.

coefficient. The regularizer $J(Q)$ measures the complexity of function Q in the function space $\mathcal{F}^{|\mathcal{A}|}$. Different choices of $\mathcal{F}^{|\mathcal{A}|}$ and J lead to different notions of complexity, e.g., various definitions of smoothness, sparsity in a dictionary, etc. As a large class of examples, $\mathcal{F}^{|\mathcal{A}|}$ could be a reproducing kernel Hilbert space (RKHS) and J^2 its corresponding norm, i.e., $J^2(Q) = \|Q\|^2_{\mathcal{H}}$.

Now suppose that, in addition to $\mathcal{D}_{\text{RL}}$, we have a set of expert examples $\mathcal{D}_{\text{E}} = \{(X_i, \pi_E(X_i))\}_{i=1}^m$, which we would like to take into account in the optimization process. The intuition behind our algorithm is that we want to use the expert examples to "shape" the value function where they are available, while using the trial-and-error data to improve the policy everywhere else. Hence, even if we have few demonstration examples, we can still obtain good generalization everywhere due to the trial-and-error data.

To incorporate the expert examples in the algorithm one might require that at the states X_i belonging to $\mathcal{D}_{\text{E}}$, the demonstrated action $\pi_E(X_i)$ be optimal, which can be expressed as a large-margin constraint: $Q(X_i, \pi_E(X_i)) - \max_{a \in \mathcal{A} \setminus \pi_E(X_i)} Q(X_i, a) \geq 1$. Nevertheless, this might not always be feasible, or desirable (if the expert itself is not optimal), so we add slack variables $\xi_i \geq 0$ to allow occasional violations of the constraints (similar to soft vs. hard margin in the large margin literature). The policy evaluation step can then be written as the following soft-constrained optimization problem:

$$\begin{aligned} \hat{Q} \leftarrow \underset{Q \in \mathcal{F}^{|\mathcal{A}|}, \xi \in \mathbb{R}^m_+}{\operatorname{argmin}} \; & \|Q - T^\pi Q\|_n^2 + \lambda_Q J^2(Q) + \frac{\alpha}{m} \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & Q(X_i, \pi_E(X_i)) - \max_{a \in \mathcal{A} \setminus \pi_E(X_i)} Q(X_i, a) \geq 1 - \xi_i. \qquad \text{for all } (X_i, \pi_E(X_i)) \in \mathcal{D}_{\text{E}} \end{aligned} \tag{2}$$

The parameter α balances the influence of the data obtained by the RL algorithm (generally by trial-and-error) vs. the expert data. When $\alpha = 0$, we obtain (1), while when $\alpha \to \infty$, we essentially solve a structured classification problem based on the expert's data [10]. In the latter case, the action-value function $\hat{Q}$ would be such that it imitates the expert demonstration data as much as possible.

Note that the above constrained optimization problem is equivalent to the following unconstrained optimization:

$$\hat{Q} \leftarrow \underset{Q \in \mathcal{F}^{|\mathcal{A}|}}{\operatorname{argmin}} \|Q - T^\pi Q\|_n^2 + \lambda_Q J^2(Q) + \frac{\alpha}{m} \sum_{i=1}^m \left(1 - \left(Q(X_i, \pi_E(X_i)) - \max_{a \in \mathcal{A} \setminus \pi_E(X_i)} Q(X_i, a)\right)\right)_+, \tag{3}$$

where $(1 - z)_+ = \max\{0, 1 - z\}$ denotes the hinge loss.

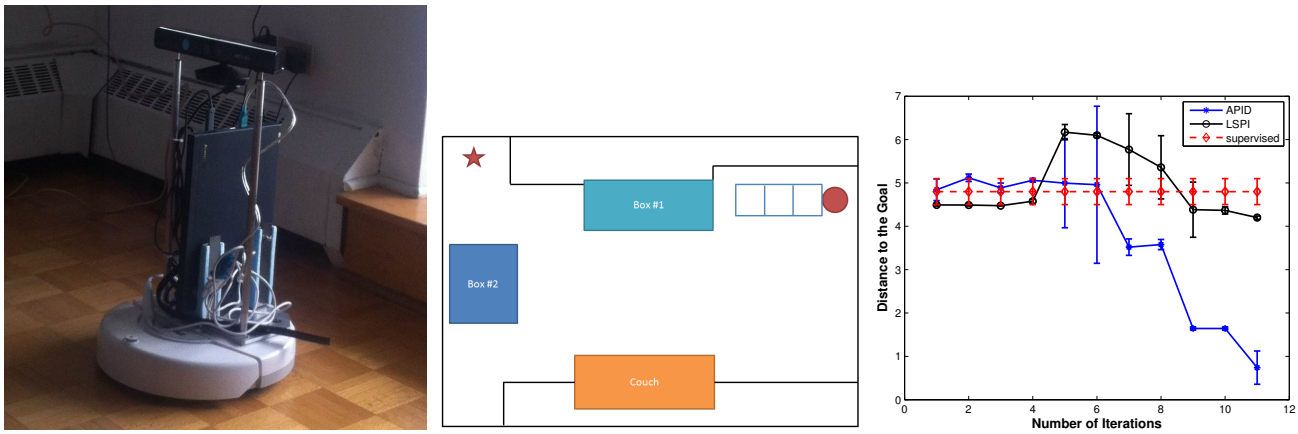
In many problems, we do not have access to the exact Bellman operator T^π, but only to samples $\mathcal{D}_{\text{RL}} = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n$ with $R_i \sim \mathcal{R}(\cdot|X_i, A_i)$ and $X_i' \sim P(\cdot|X_i, A_i)$. In this case, one might want to use the empirical Bellman error $\|Q - \hat{T}^\pi Q\|_n^2$ (with $(\hat{T}^\pi Q)(X_i, A_i) \triangleq R_i + \gamma Q(X_i', \pi(X_i'))$ for $1 \leq i \leq n$) instead of $\|Q - T^\pi Q\|_n^2$. It is known, however, that this is a biased estimate of the Bellman error, and does not lead to proper solutions [11]. One approach to address this issue is to use the modified Bellman error [11]. Another approach is to use Projected Bellman error, which leads to an LSTD-like algorithm [2]. Using the latter idea, we formulate our optimization as:

$$\begin{aligned} \hat{Q} \leftarrow \underset{Q \in \mathcal{F}^{|\mathcal{A}|}, \xi \in \mathbb{R}^m_+}{\operatorname{argmin}} \; & \left\|Q - \hat{h}_Q\right\|_n^2 + \lambda_Q J^2(Q) + \frac{\alpha}{m} \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & \hat{h}_Q = \underset{h \in \mathcal{F}^{|\mathcal{A}|}}{\operatorname{argmin}} \left[\left\|h - \hat{T}^\pi Q\right\|_n^2 + \lambda_h J^2(h)\right] \\ & Q(X_i, \pi_E(X_i)) - \max_{a \in \mathcal{A} \setminus \pi_E(X_i)} Q(X_i, a) \geq 1 - \xi_i. \qquad \text{for all } (X_i, \pi_E(X_i)) \in \mathcal{D}_{\text{E}} \end{aligned} \tag{4}$$

Here $\lambda_h > 0$ is the regularization coefficient for $\hat{h}_Q$, which might be different from λ_Q. For some choices of the function space $\mathcal{F}^{|\mathcal{A}|}$ and the regularizer J, the estimate $\hat{h}_Q$ can be found in closed-form. For example, one can use linear function approximators $h(\cdot) = \boldsymbol{\phi}(\cdot)^\top \boldsymbol{u}$ and $Q(\cdot) = \boldsymbol{\phi}(\cdot)^\top \boldsymbol{w}$ where $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{R}^p$ are parameter vectors and $\boldsymbol{\phi}(\cdot) \in \mathbb{R}^p$ is a vector of p linearly independent basis functions defined over the space of state-action pairs. Using simple l_2-regularization, $J^2(h) = \boldsymbol{u}^\top \boldsymbol{u}$ and $J^2(Q) = \boldsymbol{w}^\top \boldsymbol{w}$, the best parameter vector $\boldsymbol{u}^*$ can be obtained as a function of $\boldsymbol{w}$ by solving a ridge regression problem:

$$\boldsymbol{u}^*(\boldsymbol{w}) = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + n\lambda_h \mathbf{I}\right)^{-1} \boldsymbol{\Phi}^\top (\boldsymbol{r} + \gamma \boldsymbol{\Phi}' \boldsymbol{w}),$$

where $\boldsymbol{\Phi}$, $\boldsymbol{\Phi}'$ and $\boldsymbol{r}$ are feature and reward matrices for the trial-and-error dataset: $\boldsymbol{\Phi} = (\boldsymbol{\phi}(Z_1), \ldots, \boldsymbol{\phi}(Z_n))^\top$, $\boldsymbol{\Phi}' = (\boldsymbol{\phi}(Z_1'), \ldots, \boldsymbol{\phi}(Z_n'))^\top$, $\boldsymbol{r} = (R_1, \ldots, R_n)^\top$, with $Z_i = (X_i, A_i)$ and $Z_i' = (X_i', \pi(X_i'))$ (for data belonging to $\mathcal{D}_{\text{RL}}$). More generally, as discussed above, we might choose the function space $\mathcal{F}^{|\mathcal{A}|}$ to be a reproducing kernel Hilbert space (RKHS)



(a) (b) (c)

Figure 1: (a) Picture of the robot. (b) Hand-drawn top-down view of the environment. The star represents the goal, circle represents the initial position, black lines indicate walls, and three grid cells represents the vicinity of Kinect. (c) Distance to the goal for LSPI, APID and Supervised learning with a random forest.

and J to be its corresponding norm, which provides the flexibility of working with a nonparametric representation while still having a closed-form solution for $\hat{h}_Q$.

The approach presented so far tackles the policy evaluation step of the API algorithm. We call our approach Approximate Policy Iteration with Demonstration (APID). This step would be alternated with greedification, as in usual API. So far we have left open the problem of how the datasets $\mathcal{D}_{\text{RL}}$ and $\mathcal{D}_{\text{E}}$ are generated. These datasets might be regenerated at each iteration, or they might be reused, depending on the availability of the expert and the environment. In practice when the expert data is rare, $\mathcal{D}_{\text{E}}$ will be a single fixed batch, but $\mathcal{D}_{\text{RL}}$ could be increased by e.g., running the most current policy (possibly with some exploration) to collect more data. The generation of these datasets is application-dependent.

Note that the values of the regularization coefficients as well as α should ideally change from iteration to iteration as a function of the number of samples as well as the value function Q^{π_k}.

3 Robot Path Finding

We evaluate APID on a real robot navigation task, when $\mathcal{D}_{\text{RL}}$ and $\mathcal{D}_{\text{E}}$ are *both* expensive to obtain. We have also conducted some experiments on a simulated domain, but we do not report those results here (cf. [7]). We compare APID with LSPI and supervised LfD (Random Forest), with small $|\mathcal{D}_E|$ and only one demonstrated trajectory. We do not assume that the expert is optimal (and/or abundant).

In this task, the robot needs to get to the goal in an unmapped environment (i.e., the robot does not know where the obstacles are). We use an iRobot Create equipped with Kinect RGB-depth sensor and a laptop. The Kinect perceptual module produces a point-cloud where each point, corresponding to a pixel in the RGB image, has horizontal, vertical, and depth coordinate information. We encode the Kinect observations with a 1×3 grid cells (1m $\times$ 1m). The robot also has three bumpers to detect a collision from the front, left, and right. Figures 1a-1b show a picture of the robot and its environment. In order to reach the goal, the robot needs to turn left to avoid a first box and wall on the right, while not turning too much, to avoid the couch. Next, the robot must turn right to avoid a second box, but make sure not to turn too much or too soon to avoid colliding with the wall or first box. Then, the robot needs to get into the hallway, turn right, and move forward to reach the goal position; the goal position is set to 6m forward and 1.5m right from the initial position.

The state is represented with 3 non-negative integer features (densities in each cell) and 2 continuous features (robot position). Densities are computed by counting the number of Kinect points in a cell. The robot has three discrete actions: turn left, turn right, and move forward. The reward is minus the distance to the goal. If the robot's front bumper is pressed and the robot moves forward, it receives a penalty equal to 2 times the current distance to the goal, and if the robot's left bumper is pressed and the robot does not turn right, and vice-versa, it again receives 2 times the current distance to the goal. The robot outputs actions at a rate of 1.7Hz. We used a linear Radial Basis Function (RBF) approximator as the function approximator architecture for the value function. To solve (4), we used CVX, a package for specifying and solving convex programs [12, 13].

We started from a single trajectory of demonstration, then incrementally added trial-and-error data while fixing the expert data. The number of data points added varied at each iteration, but the average was 160 data points, which is around 1.6 minutes of exploration, gathered using an ϵ-greedy exploration policy (decreasing ϵ over iteration). Over the 11 iterations, training time was approximately 18 minutes. Initially, $\frac{\alpha}{m}$ was set to 0.9, then decreased as new data was acquired. To evaluate the performance of each algorithm, we ran each iteration's policy for a task horizon of 100 ($\sim$1 min.), and repeated 5 times, to compute the mean and standard deviation.

As seen in Figure 1c, APID outperformed both LSPI and supervised LfD. The supervised LfD method kept running into the couch. Its poor performance is due to the difference in state distribution induced by the expert and the one induced by the agent's policy [14]. LSPI had a problem of exploring unnecessary states - when ϵ-greedy exploration policy was used, it explored regions of state space that are not relevant in learning the optimal plan, such as exploring the far left areas from the initial position. APID was able to leverage the expert data to efficiently explore most relevant states and avoid unnecessary collisions. For example, it learned to avoid the first box in the first iteration, then explored states near the couch where Supervised LfD failed. Finally, Table 1 gives the time it took for the robot to get to the goal (within 1.5m). The goal was reached only in the initial expert demonstration trajectory, and in iterations 9, 10 and 11 of APID. Note that the times achieved by APID (iteration 11) are similar to the expert

Table 1: Average time to reach the goal

Average Vals	**Demonstration**	**APID-9th**	**APID-10th**	**APID-11th**
Time To Goal(s)	35.9	38.4 ± 0.81	37.7 ± 0.84	36.1 ± 0.24

4 Conclusion

We proposed a regularized algorithm that allows us to benefit from expert's demonstrations in the reinforcement learning framework. This leads to policies that perform very well even with a few data samples and gradually improve when more trial-and-error samples are collected. This extension increases the range of real-world sequential decision-making problems that can efficiently be solved. In future work, we will explore more applications of APID.

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[2] A.-m. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized policy iteration. In *NIPS 21*. 2009.

[3] G. Taylor and R. Parr. Kernelized value function approximation for reinforcement learning. In *ICML*, 2009.

[4] J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *ICML*, 2009.

[5] M. Ghavamzadeh, A. Lazaric, R. Munos, and M. Hoffman. Finite-sample analysis of Lasso-TD. In *ICML*, 2011.

[6] A.-m. Farahmand and Cs. Szepesvári. Model selection in reinforcement learning. *Machine Learning Journal*, 85(3): 299–332, 2011.

[7] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[8] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[9] D. P. Bertsekas. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, 2011.

[10] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, and Y. Singer. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

[11] A. Antos, Cs. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71:89–129, 2008.

[12] M. C. Grant and S. P. Boyd. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[13] CVX Research, Inc. CVX: Matlab software for disciplined convex programming, version 2.0. http://cvxr.com/cvx, August 2012.

[14] S. Ross, G. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Modeling active learning decisions during causal learning

Anna Coenen
Department of Psychology
New York University
New York, NY 10003
anna.coenen@nyu.edu

Bob Rehder
Department of Psychology
New York University
New York, NY 10003
bob.rehder@nyu.edu

Todd M. Gureckis
Department of Psychology
New York University
New York, NY 10003
todd.gureckis@nyu.edu

Abstract

An important type of decision making concerns how people choose to gather information which reduces their uncertainty about the world. For example, when learning about a novel piece of technology, like a smartphone, people often actively intervene on various aspects in order to better understand the function of the system. Interventions allow us to tell apart causal structures that are indistinguishable through observation, but only if the right variables are intervened on. Normative models of decision making developed in the machine learning literature specify a process of comparing hypotheses to identify those interventions that will allow a learner to distinguish between them. An experiment that asked subjects to decide between two causal hypotheses found that while they often chose useful interventions, they frequently perform interventions whose expected effects were typical of one causal structure but that did not always allow the two structures to be distinguished. We interpret this tendency as a type of positive-test-strategy with a preference for outcomes that are representative of a single causal structure.

Keywords: active learning; causal learning; interventions; information search

Introduction

To learn about causal relationships in the world, we often cannot rely on passive observation (i.e., unsupervised learning) alone. In order to understand why certain variables covary, we need the ability to actively *change* them and observe the effects of these changes. *Active interventions* are thus a crucial instrument for learning what causal structures underlie patterns of covariation in the world. There exists considerable evidence in psychology that people understand how causal systems behave in response to interventions (Waldmann & Hagmayer, 2005) and that they can use the information obtained from interventions to improve their inferences (Lagnado & Sloman, 2006).

It is still an open question, however, what strategies people use to plan their interventions with the goal of learning, that is how they decide which information would be useful for learning how a causal system works. A medical researcher, for example, needs to decide which of a patient's symptoms to treat in order to find out what illness may have caused their particular pattern of symptoms. Similarly, a scientist has to choose experimental manipulations that will tell apart different scientific hypotheses.

Here, we will examine *two* broad categories of models that can be used to explain people's decision-making processes during hypothesis testing. Then, in a behavioral experiment with human participants, we evaluate which class of models provides the best account of human decision making. The following section will give a short overview of these two key modeling approaches we have explored.

Comparative strategies

The first type of strategy that might underlie people's causal intervention decisions is based on a rational analysis of the structure learning task. According to this rational perspective, people should choose interventions that will be useful for distinguishing alternative hypotheses. There exists a larger group of optimal models, or sampling norms, that have been proposed as methods for achieving this goal (Nelson, 2005). These models share the assumption that people anticipate possible outcomes of their search behavior (here: of their interventions), and evaluate how useful these outcomes will be for differentiating hypotheses. Importantly, they all rely on a process of *comparison*, because they only value information that can help tell apart different hypotheses.

One sampling norm that captures the goal of causal structure learning particularly well is the *Information Gain* model of hypothesis testing (IG). The model aims at reducing a learner's uncertainty about which out of a number of possible hypotheses is most likely to underlie some observed data. It was first applied to causal interventions in the machine-learning literature (Murphy, 2001; Tong & Koller, 2001). However, it has also been proposed as a mechanism that guides people's intervention choices (Steyvers, Tenenbaum, Wagenmakers, & Blum, 2003).

Non-comparative strategies

In contrast to such comparative models of intervention choice, there also exists a long tradition within the psychology literature which shows that people seek information that only pertains to *one specific* hypothesis at a time. For example, it has been shown in rule-learning tasks that behavior often follows a positive-test-strategy (PTS) or positivity bias. This bias is a preference for seeking affirming information given a currently held hypothesis (e.g., Klayman & Ha, 1989), rather than testing whether the rule does not hold for counterexamples.

In the causal domain, PTS could manifest in a preference to intervene on variables (*nodes* in a causal graph), with high *centrality* (e.g., Ahn, Kim, Lassaline, & Dennis, 2000) within one candidate causal structure, irrespective of other hypotheses. Nodes are central if they have a large number of direct or indirect descendant links which could be activated through an intervention and thus count as positive evidence for a given structure. This metric can be completely at odds with a comparative strategy such as IG, because the outcomes of interventions based on this strategy may not be at all helpful for distinguishing one hypothesis from its alternatives.

Goals of this study

The aim of our study is to evaluate the degree to which people engage in comparative or non-comparative search behavior during causal structure learning. To answer this question, we conducted a simple intervention experiment that was set up to facilitate the use of a comparative strategy.

Methods

In this experiment, participants were repeatedly asked to make interventions on three-node causal systems to distinguish between two causal hypotheses.

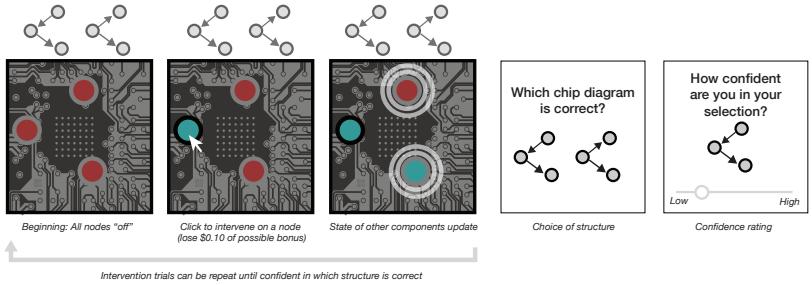


Figure 1: Intervention phase of the experiment which was repeated for each of the 27 structure comparisons. The true underlying causal graph was selected randomly. Participants could make as many interventions as they wished, but lost $0.10 of a potential bonus payment with each intervention.

Participants. We recruited one hundred and five participants (51 women and 55 men) aged 18 to 64 (M = 34.3 years, SD = 12) via Amazon Mechanical Turk. All participants were paid $2 for participation with the option of earning another $1 bonus for their performance in the task (bonus structure is explained below).

Stimuli and Materials. All possible three-node structures with one or two links were used in the experiment. They were exhaustively paired with each other to yield 27 unique structure pairs, which acted as hypotheses. All links had causal strengths of 0.8 and there were no background causes that could turn on nodes without any causal impact from another node or an intervention from the outside. In the experiment, causal graphs were described as computer chips with multiple components (nodes), which could either be on or off as indicated by their color (red or green). Hypotheses (pairs of causal graphs) were illustrated by arrow diagrams that show their respective causal links. During the task, the order of the nodes was randomized on the screen so that each node could appear in one of five different locations.

Procedure. Participants played a game which had them imagine they were working in a computer chip factory in which an accident had caused some of the chips to be mixed up. They were instructed to help identify the types of individual chips by testing them through interventions. After an extensive instruction phase, participants tested 27 chips corresponding to all 27 causal structure comparisons. They were told that each chip could be described by one of two different chip types (hypotheses), which were presented to them with arrow diagrams. The diagrams remained at the top of the screen the entire time that a chip was tested to facilitate comparison between them. For each chip comparison, one of the hypotheses was randomly selected to be the true underlying structure of the test chip.

Figure 1 illustrates the intervention phase of the experiment. Interventions could be made by clicking on one of the nodes, which could then activate other nodes on the chip. Activated nodes changed their color (from red to green). Participants could make as many interventions as they wished, and were allowed to proceed any time when they felt they had figured out the chip type. They then indicated which of the two hypotheses was most likely given the results of their interventions.

Participants could receive a bonus of up to 1$ based on one randomly chosen comparison at the end of the experiment. The bonus was only paid if they chose the correct structure at the end of that particular comparison, and it was further reduced by $0.10 for every intervention they had made. Thus, participants were incentivized to respond accurately and to use a minimal number of interventions.

Results

Model comparison.

To examine the degree to which participants rely on a comparative strategy when choosing their interventions, we calculated the expected information gain for every intervention in all 27 structure comparisons. We fit these predictions of the IG model to participants' choices using a probabilistic choice rule with a temperature scaling parameter that was estimated for each individual participant. We found that IG predicted choices well on some problem types but also considerably deviated from them on others. To make sure that these deviations were not due to just random variation, we compared bootstrapped samples from the choice data to samples from the model's posterior, separately for each problem type (plots are not shown in the interest of space). This gave us an indication of the expected uncertainty around our measurement of people's preferences, as well as the expected distribution of choices that a population of IG users would produce. Even after accounting for uncertainty in this way, model predictions from IG still deviated from the empirical

data because the two sets of samples overlapped only barely or not at all on certain problem types. We conducted the same analysis using the PTS model and found similar results (good fit on some, but not all problem types).

Next, we investigated whether a propensity for non-comparative hypothesis testing, like PTS, could explain why IG did not match people's choices in some problems. To do so, we derived a measure of agreement between the two models, by calculating the rank correlation of their predictions for the preference over the three nodes in a given problem type. Figure 2 shows how this measure of model agreement relates to the goodness of fit of the IG model, in each problem type. Indeed, we find that the IG model had a lower likelihood in precisely those problem types in which its predictions conflicted with the PTS model. In addition to the bootstrapping analyses, this provides another reason to believe that deviations from IG on some problem types are not just due to random variation in the data. Instead, the model might particularly struggle on problems where other aspects of the task, like non-comparative considerations, enter people's decision process.

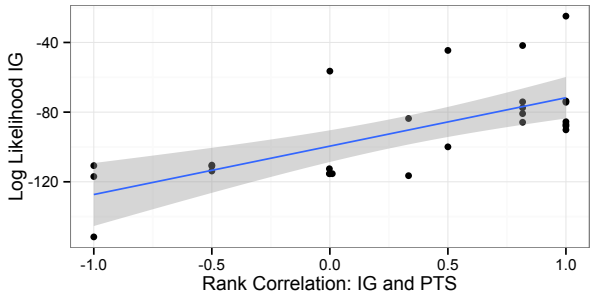


Figure 2: Log likelihood of IG model and agreement of IG and PTS (kendall's tau rank correlation), by problem type.

Finally, we fit a combined model that took a weighted combination of IG and PTS scores before applying the probabilistic choice rule. Again, weights were estimated separately for each participant. When comparing posterior samples of this combined model to bootstrapped samples of the data, we found that it made credible predictions on all 27 structure comparisons.

Reaction times.

If, as the combined model suggests, participants are influenced by both comparative and non-comparative aspects of the task, we expected that it should be particularly difficult to choose an intervention when IG and PTS make divergent predictions about which node to choose. We therefore looked at the time it took participants to make an intervention, separately for each problem type and again depending on the agreement between IG and PTS. As Figure 3 shows, people did take significantly longer to choose an intervention in problems with low model agreement, $r(25) = -0.58$, $p < 0.005$. This finding confirms that comparative and non-comparative components may both play a role in people's intervention decisions and, when in conflict, can make certain problems more difficult than others.

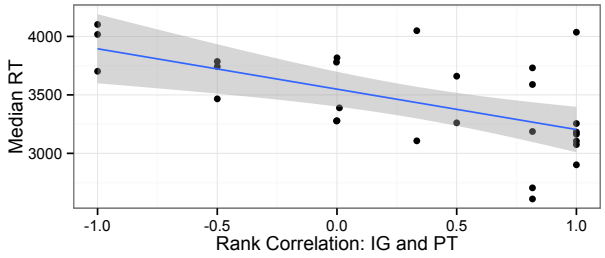


Figure 3: Median response time before making an intervention and agreement of IG and PTS (kendall's tau rank correlation), by problem type.

Individual differences

Using the combined model, we found considerable variation in the relative weights that participants place on the two strategies examined here (IG and PTS). Thus, we were interested in finding out if the individual tendency to use either IG or PTS manifested in other aspects of participants' behavior in the task, besides their intervention choices. To do so, we considered the difference in log likelihood of the separate IG and PTS models for each participant as a proxy for their tendency of making comparison-based interventions. We considered three independent variables in relation to this measure:

First, we predicted that participants who are more prone to using IG, which is a computationally more intensive strategy than PTS, would take longer to decide which intervention to make. As predicted, we find that participants whose behavior is better accounted for by the IG model compared to PTS take significantly longer to choose interventions, $r(103) = 0.21$, $p = 0.03$.

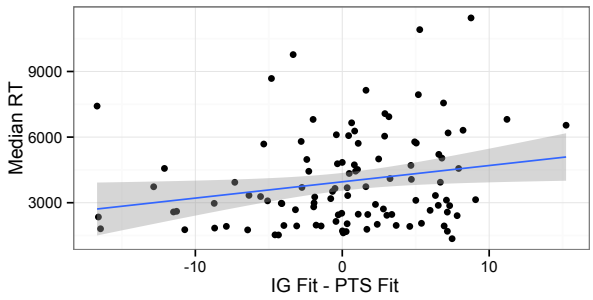


Figure 4: Response time and difference in model fit between IG and PTS, by participant.

We also expected that IG users would be more likely to choose the correct causal structure at the end of the intervention phase. This is plausible because using IG leads to outcomes that will allow the learner to actually discriminate between

graphs. It is also possible that if a non-comparative strategy is used, learners are more likely to falsely rely on outcomes that appear to provide evidence for one of the graphs, but in fact do not exclude the possibility that the alternative is true.

As expected, we found a positive relationship between the degree to which participants' choices were better fit by the IG model and their average accuracy across all comparisons, $r(103) = 0.28,\ p < 0.01$, as shown in Figure 5.

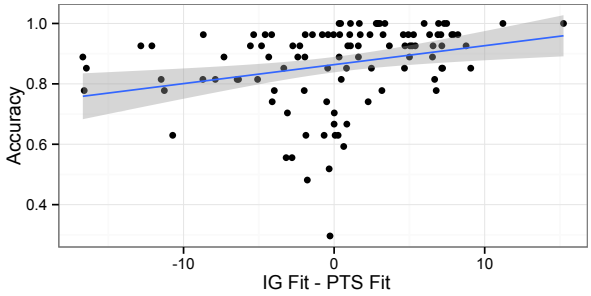


Figure 5: Accuracy and difference in model fit between IG and PTS, by participant.

Finally, we also expected comparative hypothesis testers to need fewer interventions overall before deciding which structure is correct. Again, one reason for this is that they should have received better data on average to help them actually discriminate the two graphs. Another reason is that positive testers might be tempted to want to recreate all positive effects of one of the structures and thus require more interventions to achieve this goal. As figure 6 shows, individuals better fit by IG made fewer interventions than participants who relied more heavily on the non-comparative strategy, $r(103) = -0.35,\ p < 0.001$.

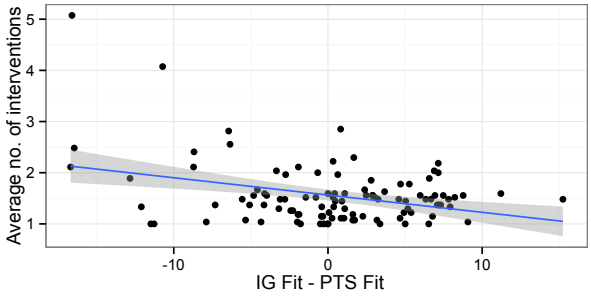


Figure 6: Number of interventions and difference in model fit between IG and PTS, by participant.

In sum, the combined model of IG and PTS not only provides a better fit to people's choices, but it also has some interesting behavioral implications that we could observe in our data.

Discussion

In contrast to predictions of the rational approach to causal information search, we find that people's intervention choices not always aim at differentiating causal hypotheses. Instead, participants' choices in a simple causal intervention task were best accounted for by a model that also included preferences based on graph-specific, non-comparative features of a given problem. Specifically, participants preferred intervening on causal nodes that had the potential to trigger a large proportion of all the effects associated with *one* of the hypothesized graphs. We interpret this preference as a type of positive-test-strategy, which favors seeking information that will lead to positive outcomes that should be expected if a given graph was true. This finding is at odds with a rational model that is purely based on seeking interventions that lead to surprising outcomes, like the IG model. In reality, it looks like people's decisions are guided by both comparative and non-comparative strategies during intervention-based causal structure learning.

Going forward, we are interested in testing whether people's reliance on non-comparative strategies can be influenced by the task environment. In our current experiment, using a non-comparative strategy still led to outcomes that would enable participants to make correct graph choices, most of the time. However, if graph comparisons were designed so that non-comparative strategies would not aid learning at all, it is possible that participants would switch to a more comparison driven approach.

References

Ahn, W.-k., Kim, N. S., Lassaline, M. E., & Dennis, M. J. (2000). Causal status as a determinant of feature centrality. *Cognitive Psychology*, *41*(4), 361–416.

Klayman, J., & Ha, Y.-w. (1989). Hypothesis testing in rule discovery: Strategy, structure, and content. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *15*(4), 596.

Lagnado, D. A., & Sloman, S. A. (2006). Time as a guide to cause. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *32*(3), 451.

Murphy, K. P. (2001). Active learning of causal bayes net structure. *Technical Report. Department of Computer Science, U.C. Berkeley.*.

Nelson, J. D. (2005). Finding useful questions: on bayesian diagnosticity, probability, impact, and information gain. *Psychological review*, *112*(4), 979.

Steyvers, M., Tenenbaum, J. B., Wagenmakers, E.-J., & Blum, B. (2003). Inferring causal networks from observations and interventions. *Cognitive science*, *27*(3), 453–489.

Tong, S., & Koller, D. (2001). Active learning for structure in bayesian networks. In *International joint conference on artificial intelligence* (Vol. 17, pp. 863–869).

Waldmann, M. R., & Hagmayer, Y. (2005). Seeing versus doing: two modes of accessing causal knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *31*(2), 216.

Modelling effects of intrinsic and extrinsic rewards on the competition between striatal learning systems

Joschka Bödecker
Department of Computer Science
Albert-Ludwigs-University Freiburg
Freiburg, D-79110, Germany
jboedeck@informatik.uni-freiburg.de

Thomas Lampe
Department of Computer Science
Albert-Ludwigs-University Freiburg
Freiburg, D-79110, Germany
tlampe@informatik.uni-freiburg.de

Martin Riedmiller
Department of Computer Science
Albert-Ludwigs-University Freiburg
Freiburg, D-79110, Germany
riedmiller@informatik.uni-freiburg.de

Abstract

A common assumption in psychology, economics, and other fields holds that higher performance will result if extrinsic rewards (such as money) are offered as an incentive. While this principle seems to work well for tasks that require the execution of the same sequence of steps over and over, with little uncertainty about the process, in other cases, especially where creative problem solving is required due to the difficulty in finding the optimal sequence of actions, external rewards can actually be detrimental to task performance. Furthermore, they have the potential to undermine an intrinsic motivation to do an otherwise interesting activity. In this work, we extend a computational model of the dorsomedial and dorsolateral striatal reinforcement learning systems to account for the effects of extrinsic and intrinsic rewards. The model assumes that the brain employs both a goal-directed and a habitual learning system, and competition between both is based on the trade-off between the cost of the reasoning process and value of information. The goal-directed system elicits internal rewards when its models of the environment improve, while the habitual system does not. We test the hypothesis that external rewards bias the competition in favour of the computationally efficient, but cruder and less flexible habitual system, which can negatively influence intrinsic motivation in the class of tasks we consider, whereas intrinsic rewards can lead to faster learning. Thereby, we account for the phenomenon that initial extrinsic reward leads to reduced activity after extinction compared to the case without any initial extrinsic rewards.

Keywords: cognitive modelling; extrinsic rewards vs intrinsic motivation; striatal competition; reinforcement learning

1 Motivation

What motivates intelligent beings to perform certain actions in their environment is a central question in psychology. The influential paradigm of operant conditioning [9] held that all behavior is stimulated by rewards presented to an animal. This view was challenged, however, by observations made by White [10] that some behaviors are *intrinsically motivated*, i.e., they are performed simply because the activity is *intrinsically rewarding*. Deci then examined what effects external rewards would have on intrinsic motivation [3] and found that under certain circumstances, extrinsic rewards could undermine intrinsic motivation. Later on, several studies (see extensive meta-analytic review in [4]) observed that external rewards can decrease cognitive flexibility in problem solving, and have the potential to decrease performance on complex tasks. These findings significantly contradicted predictions of earlier theories such as operant conditioning or utility theory in economics.

To explain these observations, several theoretical accounts have been put forward (e.g. Self-Determination Theory [7] amongst others) which suggest different cognitive mechanisms to account for the data. However, it is not clear what *computational* mechanisms give rise to these phenomena. A computational model would enable quantitative comparisons of different hypotheses, test various experimental settings, and generate predictions for new, untested scenarios.

Here, we provide such a computational model by extending two previously presented models explaining behavioral control in the striatal systems [1], and trade-offs between habitual and goal-directed brain processes [5]. Both of these models follow a hypothesis from behavioral economics, suggesting that two distinct control systems in the brain compete for control of actions. The models are formalized using the framework of reinforcement learning (RL), and it is assumed that one controller uses computationally efficient model-free RL, and the other one uses statistically efficient model-based RL algorithms. The model-free system models a habitual process, implementing a cache of efficient actions for a given situation, while the model-based system realizes a goal-directed process by searching a tree of recorded state-action transition probabilities for alternative choices. Both computational models could account for several phenomena from animal experiments designed to test devaluation resistance, including habituation after extensive training, non-habituation in ambivalent tasks, and habituation in preference tasks. Our proposed model is a mixture of both earlier models (see below for details), and extends them by including intrinsic rewards for the model-based goal-directed subsystem. With this extension, we aim to explain two additional phenomena which the previous models could not account for.

Reduced post-extinction activity – In creative tasks, the presence of strong extrinsic rewards can lead to diminished activity after said rewards have been devalued. More specifically, the activity will be lower than it would have been had the subject never received any extrinsic reward in the first place [3]. Strong extrinsic rewards are therefore expected to suppress intrinsic motivation.

Activity without extrinsic reward – When dealing with a creative or complex system, both humans and animals can be observed to interact (to ”play”) with it even if no extrinsic reward whatsoever is being provided or promised.

2 Related Work

Both previous models [1, 5] intend to give a formal account of the striatal system and its division in a model-based ”tree” module and a model-free ”cache” module. They argue that the observed effects are caused by the adaptable model-based system being active initially, but being replaced by the less adaptive but cheaper model-free system after extended training in a devaluation task. The main difference lies in the competition mechanism used to arbitrate between both systems in each model.

Uncertainty-based competition – In the earlier model by Daw et al. [1], it is assumed that the system is chosen which is more certain about the action to be taken. To determine uncertainty, both the model-based and the model-free system are implemented using Bayesian Learning [2]. Therefore, rather than learning Q-values for a given state, they learn distributions over them, meaning that each entry in the Q-table is represented by a mean and variance. Likewise, the transition function and the terminal reward function employed by the model-based subsystem are also tables of distributions that are adapted as experience is accumulated. A policy is then generated through tree search on this model. During action selection, each available action's Q-value to be used for exploration is then provided by the system with the lower variance.

Since the tree search is performed until all the way to its leaf nodes rather than just along local edges, a sudden change in the reward model resulting from a devaluation event will immediately be propagated all the way through the state space. In contrast, the model-free system will have to perform the original sequence several times to register a change in the terminal state's value in the starting state.

Value-based competition – Keramati et al. [5] adapt the basic approach of Daw et al. [1] to use the value of perfect information (VPI) instead of uncertainty. Here, the model-free system computes how much value would be gained from knowing the true value of a given action. Intuitively, this value is higher if an action's Q-distribution overlaps strongly

with the best action, since in this case the former may turn out to be preferable. Conversely, once the distributions have separated, knowing the true value of an action is unlikely to change which one is ultimately chosen. The VPI is then compared against the costs of opportunity for performing a tree search, denoted by $\bar{R}\tau$, with $\bar{R}$ being the average reward observed thus far and τ being the cost in terms of deliberation time for traversing an edge of the tree. Only if the VPI is higher than the opportunity costs is the model-based system activated to determine the true reward, which is then used for action selection.

3 Theory Overview

The model we describe is primarily an extension of the one proposed by Daw et al. [1]. It also adopts some, but not all, of the revisions introduced by Keramati et al. [5], so that our model can be considered to be a mixture between both.

Like in the latter, we use the VPI to mediate between the goal-directed and the habitual subsystem. The alternative approach of using the variance of the Q-function's estimates would not be plausible in a framework containing intrinsic rewards. Intrinsic motivation is generally assumed to be high for regions of the state space in which the model has not been learned yet. In these regions, the goal-directed system's variance will also be particularly high. If the goal-directed system's variance is involved in the competition mechanism, this will lead to it being rejected in precisely those situations when intrinsic motivation is high, thereby neutralizing the effect of the latter.

From the original approach by Daw et al. [1] we retain the use of Beta and Dirichlet distributions to represent the model and the policies learned by the agent. Using Beta distributions for the policy carries the advantage of being able to represent a limited amount of ambiguity arising from non-determinism, while Gaussians only model uncertainty.

Beyond these adaptations, there are two major extensions in our model that were not present in its predecessors, which will be described in detail in the following.

Intrinsic rewards – The main contribution of our model lies in its extension with a mechanism for intrinsic motivation. Currently we consider only one of multiple types of intrinsic reward, namely the learning progress of the transition model. There are other proposed aspects to intrinsic motivation, such as competence-based and information-theoretical mechanisms (for an overview, see e.g. [6]), but we focus on progress for the sake of simplicity, as it alone already accounts for the phenomena we consider.

The central feature of intrinsic rewards lies in that their value depends on the current state of the model, as opposed to extrinsic rewards that are provided by the process or environment. As such, intrinsic rewards can notably arise *only* in the goal-directed system, and are not applied to the habitual one.

As measure of learning progress we use the magnitude of shifts in the means of the transition function's distributions. Formally, the intrinsic reward $I_{s,a}$ for choosing action a in state s is given by the equation:

$$I_{s,a} = \sum_{s' \in S} |\Delta \mu^{trans}_{s,a,s'}| \tag{1}$$

This value is, together with the transition costs, added to the result of the tree search:

$$\hat{Q}_{tree} s, a := Q^{tree}_{s,a} + I_{s,a} \tag{2}$$

The resulting Q-values $\hat{Q}_{tree}$ are then used in place of those determined by the search for the purpose of action selection.

Transition costs – Aside from intrinsic rewards, we also introduce transition costs. While a common element of reinforcement learning and formalized in the Bellman Equation, they were not present in the model by Daw et al. [1]. Instead, the terminal extrinsic reward of a trajectory was propagated all the way to the starting state. By accomodating them, we enable the model to acquire minimum-time policies in tasks where trajectories can contain loops. Transition costs can also be chosen differently for each action, thereby modelling energy conservation.

It is worth noting that action-based transition costs do not fall cleanly into the distinction between extrinsic and intrinsic rewards. Traditionally considered extrinsic rewards, they are likewise applied to the habitual system, as opposed to intrinsic rewards, which due to being model-based can naturally only occur within the goal-directed system. On the other hand, they mimic intrinsic rewards in that they are essentially inherent – one may be tempted to say "intrinsic" – to the agent. Action costs are not provided by the environment, and can thus be assumed to occur even when other extrinsic rewards do not.

Applying transition costs can easily be done by adding them to the target mean derived from the successor state during both tree search and update of the habitual system, yielding a target mean $\hat{\mu}$:

$$\hat{\mu}_{s,a} = \mu_{s',a_*} + r_a \tag{3}$$

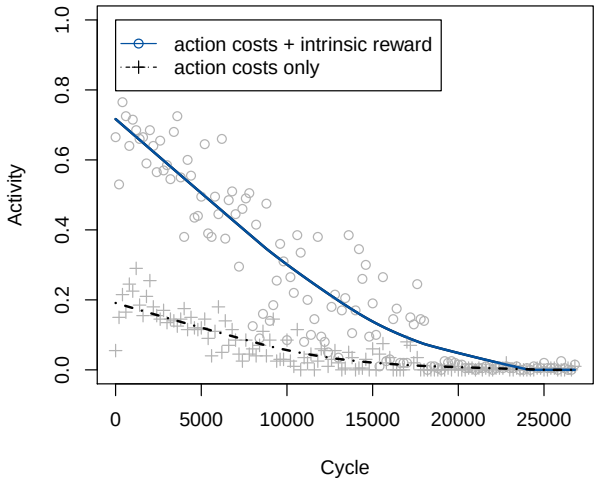


Figure 1: Development of the percentage of non-null action choices, with and without intrinsic rewards. Curves are based on the theoretical greedy choice of action, even in the 20% of cycles in which an ϵ-greedy exploratory action was ultimately used. Ratios were determined across bins of 200 samples and smoothed using locally weighted scatterplot smoothing.

Since the update rule for the distribution parameters also requires the second moments of the successor states' Beta distributions, we generate a new distribution $\hat{Q}_{s,a} = Beta(\hat{\alpha}, \hat{\beta})$ with the target mean $\hat{\mu}_{s,a}$.

4 Experimental Results

Our predecessor models [1, 5] were primarily examined using a simple decision task inspired by experiments with rats, where the animals had to manipulate a feeding apparatus in a short sequence to generate an extrinsic reward. However, these tasks consist only of very few states and actions, making them too simple to showcase those phenomena related specifically to intrinsic motivation. We therefore consider a more complex setting, adapted from the Playroom environment used by Singh et al. [8], albeit simplified to accommodate the use of classical Bayesian RL.

In this task, the agent has to learn to manipulate a number of objects, each of which causes a different effect when used. The agent possesses a hand and an eye, both of which must rest on an object for it to become usable. Aside from performing an object affordance, the agent can also move its eye to a random object, bring the hand to the object the eye is resting on, or perform a null action that has no effect whatsoever. The null action generates a small action reward, unlike the other actions which cause negative ones. We thereby model an agent's general tendency to prefer the action that exerts the least effort. While still simple for a task aimed at intrinsic motivation, it is considerably more complex than the food dispensal experiments. Most notably, trajectories can be cyclic, and one of the actions is non-deterministic. In addition, the partial observability of the state when the light is off can lead to local minima in the policy. In this framework, we observe the behavior of the system using different combinations of intrinsic and extrinsic rewards whether the phenomena described in section 1 can be reproduced.

Activity without extrinsic rewards – A first experiment compares the activity of the system with and without intrinsic rewards. In this setting, there are no external rewards whatsoever, aside from the action-dependent transition costs. One would expect the overall activity, i.e. the occurrence of non-null actions, to be increased when not using intrinsic rewards. And indeed, as Figure 1 illustrates, their use leads to a significantly lower rate at which the null action is chosen. The activity with intrinsic motivation drops to a similar level as without it much later, once the model has stabilized and no more intrinsic reward can be generated.

Post-extinction activity – To show that stronger extrinsic rewards lead to less activity, as hypothesized in section 1, we next have the system learn a policy while providing the maximum extrinsic reward upon entering the goal state s_+. In this case, s_+ is reached by having the music turned on and the lights off. After 200 episodes of training in the late-evaluation case, and 75 episodes in the early devaluation case, we devalue it by replacing the distribution of the extrinsic reward model for the goal state with the Beta distribution $Beta(1, 15)$. The parameters of the replacement distribution were chosen in accordance with Daw et al. [1] in such a way as to concentrate most of the probability mass at 0.

As one would expect, the post-devaluation activity, shown in Figure 2(a), drops sharply compared to its earlier level. While the habitual system remains active, being unable to adapt to the change immediately, the goal-directed system immediately switches to the use of the null action.

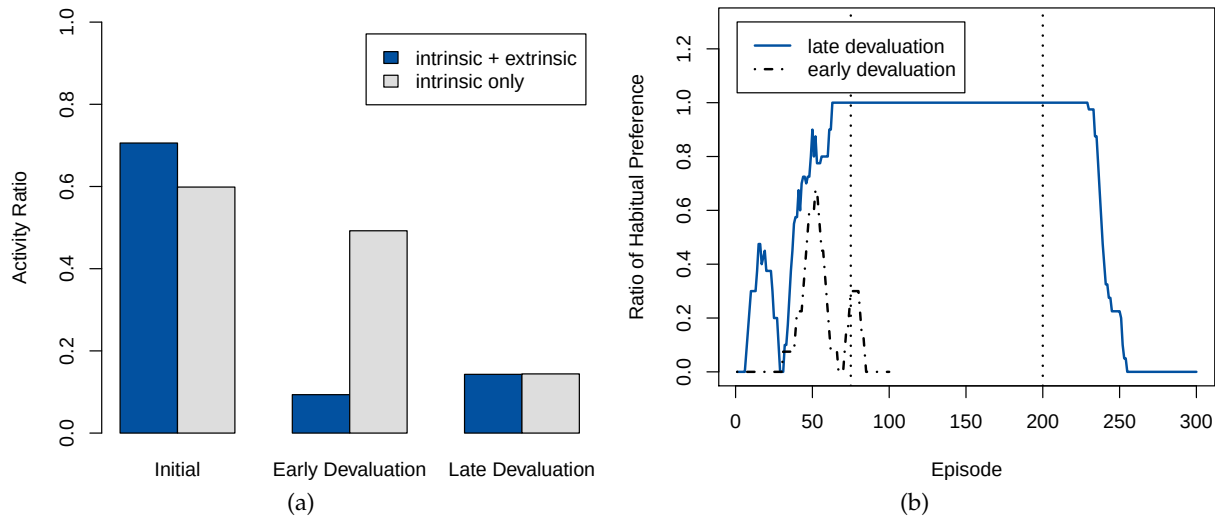


Figure 2: Behavior of the system when using both intrinsic and extrinsic rewards, with devaluation occurring at episode 75 or 200. (a) Percentage of non-null actions chosen before and after devaluation, as well as during a run without extrinsic rewards. (b) Ratio of how often the habitual system is selected. The vertical line marks the time of devaluation.

As the costs of opportunity for performing a tree-search decrease, it takes over from the habitual system as seen in Figure 2(b). The previous takeover of the habitual system caused the agent to be active mostly in a limited region of the state space, as any exploration attempts were cut short by the habitual system's drive to reach the goal. Consequently, the model in this area of the state space is very accurate already. Therefore, no intrinsic reward is generated anymore, and the goal-directed system will not deviate from its path once having taken over. Essentially, due to the prolonged activation of the habitual system, the intrinsic motivation will have been exhausted without having the chance to cause any increased exploration and activity.

Most importantly, the activity after extinction is significantly lower than that which would result from using no extrinsic rewards in the first place, as illustrated in Figure 2(a). This effect is caused by the model having stabilized along the trajectory learned and continuously repeated by the habitual system. Therefore, no intrinsic reward is generated anymore, and the goal-directed system will not deviate from its path once having taken over. Also note that the purely intrinsic setting results in slightly lower activity than the pre-devaluation case. This seems plausible, since a system not driven by extrinsic rewards would be more likely to try the sub-optimal null action to improve its model.

References

[1] N. D. Daw, Y. Niv, and P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711, 2005.

[2] R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI)*, AAAI/IAAI, pages 761–768, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[3] E. L. Deci. Effects of externally mediated rewards on intrinsic motivation. *Journal of Personality and Social Psychology*, 18(1):105–115, 1971.

[4] E. L. Deci, R. Koestner, and R. M. Ryan. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological Bulletin*, 125(6):627–668, 1999.

[5] M. Keramati, A. Dezfouli, and P. Piray. Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLoS Computational Biology*, 7(5):e1002055, 2011.

[6] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1(6), 2007.

[7] R. M. Ryan and E. L. Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55(1):68–78, 2000.

[8] S. Singh, A.G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17: Proceedings of the 2004 Conference*, Cambridge, MA, 2005. The MIT Press.

[9] B. F. Skinner. *Science and human behavior*. Macmillan, New York, NY, USA, 1953.

[10] R. W. White. Motivation reconsidered. *Psychological Review*, 66:297–333, 1959.

Stimulus detection and decision making via spike-based reinforcement learning

Giancarlo La Camera
Department of Neurobiology and Behavior
Stony Brook University
Stony Brook, NY 11794, USA
giancarlo.lacamera@stonybrook.edu

Robert Urbanczik
Department of Physiology
University of Bern
Bühlplatz 5, Bern, Switzerland
urbanczik@pyl.unibe.ch

Walter Senn
Department of Physiology
University of Bern
Bühlplatz 5, Bern, Switzerland
senn@pyl.unibe.ch

Abstract

In theoretical and experimental investigations of decision-making, the main task has typically been one of classification, wherein the relevant stimuli cueing decisions are known to the decision maker: the latter knows which stimuli are relevant, and knows when it is being presented with one. However, in many real-life situations it is not clear which segments in a continuous sensory stream are action relevant, and relevant segments may blend seamlessly into irrelevant ones. Then the decision problem is just as much about when to act as about choosing the right action. Here, we present a spiking neuron network which learns to classify hidden relevant segments of a continuous sensory stream of spatio-temporal patterns of spike trains. The network has no a-priori knowledge of the stimuli, when they are being presented, and their behavioral significance – i.e., whether or not they are action-relevant. The network is trained by the reward received for taking correct decisions in the presence of relevant stimuli. Simulation results show that by maximizing expected reward the spiking network learns to distinguish behaviourally relevant segments in the input stream from irrelevant ones, performing a task akin to temporal stimulus segmentation.

Keywords: spiking neuron; temporal segmentation; signal detection; gradient learning; synaptic plasticity; spike-timing patterns; firing rate patterns; neural circuit

Acknowledgements

We acknowledge support from the National Science Foundation (Grant IIS-1161852) and the Swiss National Science Foundation through the SystemsX.ch initiative ("Neurochoice").

Spiking network models aspire to produce biologically plausible models of learning and decision making (see e.g. [1]). For concreteness, consider the following 2 choice classification task: a set of input stimuli is to be associated with one of two possible correct actions – e.g., 'go left' vs. 'go right'. The correct decision is rewarded whereas the incorrect decision is punished. In a 'canonical' spiking network model designed to learn this task [2, 3], populations of sensory neurons project to populations of 'decision neurons' via plastic synapses, as shown in Fig. 1a. Each stimulus is represented by the activation of a predefined sensory population, such as the orange population in Fig. 1a. After an input is presented to the network, some competition occurs at the level of the decision populations, which ends when one of the two populations enters a state of activity having higher firing rate than the other (or, in alternative models, its activity reaches a pre-defined threshold earlier than the other population). The winning population initiates the corresponding action. If that action is correct, the network is rewarded, otherwise it is punished. Based on this outcome, the synapses between the input neurons and the decision neurons are modified so as to increase the chance of producing the correct action in response to future presentations of the same stimulus. This class of models are able to capture much of the physiology and behavior observed in typical laboratory tasks which inspired them [1]; however, they are designed to work in a somewhat limited scenario, in which: 1) every stimulus presented to the agent is *relevant*, in the sense that, if met with the correct action, a reward is obtained; 2) the agent knows the identity of all the stimuli and when they are being presented; 3) there is a well-defined time period during which a decision must be made (decisions are enforced); 4) all decisions lead to feedback (either reward or punishment) – hence, feedback is received for each stimulus presentation. Also, the network model of Fig. 1a has as many input populations as relevant stimuli: to introduce a new stimulus, one has to augment the model with an additional population of neurons encoding that stimulus.

Here, we present a spiking network model (the learning agent, or 'agent' for short) which learns to segment a continuous input stream by identifying those segments of the stream that are action-relevant (see Fig. 1b). The relevant stimuli are spatio-temporal patterns of spike trains hidden among a host of non-relevant patterns in the same continuous stream. Learning is achieved with an online, spike-based learning rule that tries to maximize reward. Compared with the learning scenario outlined above, here 1) the a-priori relevance of the stimuli is not known to the agent; 2) the agent does not know when and if a stimulus is being presented; 3) the agent is *not required* to make a decision at any time; and 4) only *correct* decisions made in the presence of a *relevant* stimulus lead to feedback. This is the fundamental distinction between relevant and non-relevant stimuli: if any decision is made in the presence of a non-relevant stimulus, nothing happens – in particular, no rewarding feedback is given. If every action is costly (as assumed below), the optimal behavior in the presence of non-relevant stimuli is to *do nothing*.[1] Finally note that, contrary to the model of Fig. 1a, in our network additional stimuli can be represented as new segments of the stream, with no need to add populations of input neurons.

Network architecture and decision dynamics. The spiking network model we propose in this work is illustrated in Fig. 1b. Two decision populations of $N = 100$ spiking neurons each (labeled as $\mathbb{L}$ and $\mathbb{R}$ respectively) receive input spike trains via plastic synapses (Fig. 1b). When the difference in spike counts between the two populations exceeds a threshold Θ_D, $|spk(L) - spk(R)| > \Theta_D$, a decision occurs. As long as $|spk(L) - spk(R)| < \Theta_D$, no decisions are taken. Each stimulus is randomly deemed either relevant or irrelevant, with relevant stimuli arbitrarily associated to one of two correct decisions, either 'go left' (accomplished if $spk(L) - spk(R) > \Theta_D$), or 'go right' (accomplished if $spk(L) - spk(R) < \Theta_D$). When a decision occurs, a rewarding feedback is obtained after a minor delay (50ms), the stimulus is removed, and the population activity is reset to zero. Every decision (whether correct or incorrect) incurs a small cost -0.1 (to prevent the agent to take decisions continuously), and positive reward ($R = 1$) is given only for a correct decision in the presence of a relevant stimulus (netting a total reward of $R = 0.9$). Incorrect decisions are not punished (and thus only incur the cost $R = -0.1$). The rationale for such choice is that an additional negative reward for an incorrect response to a relevant stimulus would signal the presence of a relevant stimulus at the time of a decision, aiding the solution of the identification task. In case of multiple correct responses to the same relevant stimulus, only the first such response is rewarded. We tested the model with both precise spike timing patterns (task 1) and firing rate patterns (task 2), as detailed in a later section. In both tasks, stimuli were of random duration around a mean of 500ms.

Decision neurons and learning rule. We indicate with P_s a smoothed version of the readout $spk(L) - spk(R)$ in the following. The neurons contributing to the population activity responsible for making decisions were modeled as spike response models with a noisy escape mechanism for action potential emission [4] – i.e., a spike is emitted with a given probability $\phi(u(t))$ depending on the current value of the membrane potential u at time t. Learning occurred via the online learning rule introduced in [5],

$$\frac{dw_i^\nu}{dt} = \eta |R_t| a(P_s)(r^\nu - 1) E_i^\nu, \tag{1}$$

where w_i^ν is the synaptic weight between pre-synaptic (input) neuron i and post-synaptic (decision) neuron ν, η is the learning rate, R_t is the reward at time t, r^ν is an individualized reward signal that equals 1 if neuron ν made the right decision, and -1 otherwise. The factor $|R_t|$ insures that synaptic update is confined to a temporal window around reward delivery. $E_i^\nu(t) \propto \left(\sum_{t^\nu} \delta(t - t^\nu) - \phi(u^\nu(t))\right) PSP_i(t)$ is a low-pass filter of the time-derivative of the gradient (with respect to the synaptic weights) of the log-likelihood of producing the output spike train $\{t^\nu\}$ given an input spike

[1] Note how this is different from a 3-way classification task where the stimuli are to be separated in 3 classes ('go left', 'go right', and 'do nothing'), and in which ignoring non-relevant stimuli would be rewarded as the correct response to those stimuli.

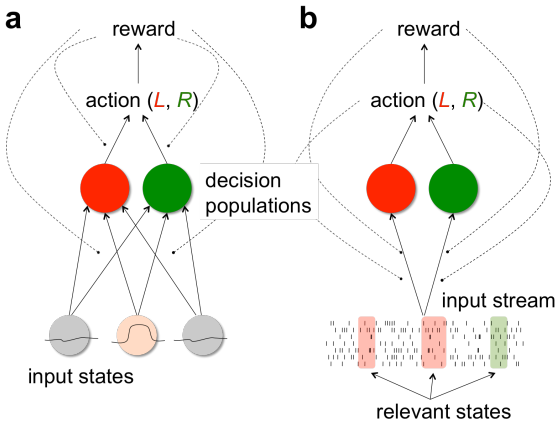


Figure 1: Alternative neural circuit models for decision making tasks. **a)** In a 'canonical' decision-making circuit, each input stimulus is represented by an increase in firing rate in a dedicated population of neurons (here, the orange population). Two decision populations code for 'go left' (red) and 'go right' (green), respectively. A read-out initiates either action, and the decision is met with a reward or punishment. The outcome modulates synaptic plasticity (dashed curves) at either one of the pathways or both. To represent a new relevant stimulus, a new population of neurons must be added to the network. **b)** In the type of cortical circuit studied in this paper, the input is a spatio-temporal pattern of spike trains (each spike train coming from a different input neuron). Relevant inputs are hidden segments of this pattern (shaded areas): if met with the appropriate response, a reward is delivered. The network has no a priori knowledge of the relevant segments: these are formed by segmenting the input through a process of reinforcement learning. No additional populations are required to represent additional stimuli – whether relevant or not. See the text for details.

pattern causing a post-synaptic potential $PSP_i(t)$ on neuron i at time t (see [4, 5] for details). Note that only the synapses targeting neurons voting for the wrong decision ($r^\nu = -1$) are updated according to the above learning rule; the update is full ($a(P_s) = 1$) in case of an incorrect decision and attenuated by a factor $a(P_s) \propto e^{-P_s^2/N}$ in case of a correct decision. This allows for synapses to undergo a full update only when most needed (i.e., following a wrong population decision). Moveover, synaptic updates for neurons voting for incorrect decisions are smaller for a larger population readout P_s because of the value of the attenuation factor $a(P_s)$ in this case. Since P_s can be interpreted as an internal measure of the agent's 'confidence' in its decision, the synaptic update is small for correct decisions taken with large confidence.

In the case of episodic learning, the learning rule Eq. 1 performs stochastic gradient ascent in a monotonic function of reward and population activity [5]. This learning rule can be understood as an improvement over Williams's general gradient learning rule [6]. The need to introduce the individualized reward signal r^ν arises because otherwise learning worsens as the population size increases, as demonstrated in [5]. The individualized reward signal can be made available locally at each synapse by broadcasting feedback from the population readout P_s (e.g., through a neurotransmitter such as acetylcholine or serotonin) and from each neuron's own activity S_t (e.g., through intracellular calcium transients), in addition to the global reward feedback R_t (see [5] for details).

Finally, learning occurred only on synapses targeting neurons in the $\mathbb{L}$ population, with the synapses projecting onto the $\mathbb{R}$ population kept fixed. This way, the $\mathbb{R}$ population was a 'contrast population' used as reference for making decisions. Since the only variable responsible for decisions is the difference between the activities of the two populations, this choice is legitimate and allows for a minimal implementation. Note that there is no *a priori* preference for which population should be the learning one: their roles can be interchanged without affecting the results.

Simulation results with spike timing patterns (task 1). In this scenario, stimuli were patterns of 60 spike trains. Each spike train was obtained as a realization of a Poisson process with a constant firing rate of 6Hz. The choice of a Poisson process is convenient but not strictly necessary, i.e., any other distribution could have been used instead [7]. Once created, the spike patterns were presented each time unmodified, i.e., for each pattern, the spike trains were kept fixed across repeated presentations ('frozen' patterns; this is an un-biological simplification that will be relaxed later). Note that all spike patterns have exactly the same statistics, and thus they cannot be encoded or decoded by firing rate. As shown in [5], stimuli of this sort can be classified by a single decision population equipped with the learning rule Eq. 1 within a so-called 'time controlled' paradigm [8], whereby an action is required at the end of stimulus presentation and no stimulus identification is involved. Here, however, relevant stimuli must be identified first, and decisions can be made at any time, or could not be made at all. A simulation run for this classification task with the online learning rule Eq. 1 is shown in Fig. 2 for the case of 6 stimuli (the same model can also learn tens of stimuli; not shown here to ease illustration). The network was able to learn to identify relevant segments and make the correct decision in response to them (pink and green shaded areas in panel a), while holding actions in the presence of non-relevant segments – at least in a large fraction of them and given the limited simulation times. Performance tends to increase with learning (panel b, top). The evolution of decision times for the best and the worst stimuli (panel b, bottom) shows that as the network

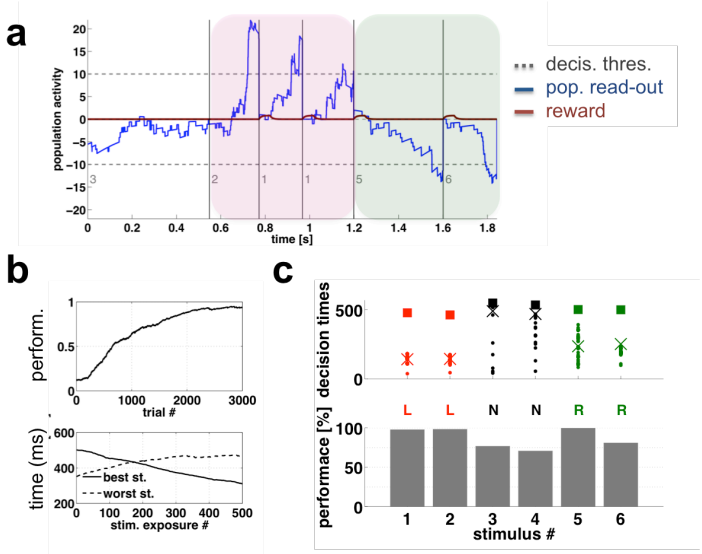


Figure 2: Simulation results with 'frozen' patterns of spike trains. **a)** Dynamics of decisions after learning for 3000 trials in the architecture of Fig. 1b. The population readout (P_s in the main text) correctly makes the decision to go left in response to the 'pink' segments of the input stream, and to go right in response to the 'green' segments, by crossing a threshold (dashed horizontal lines, positive for decision 'left' and negative for decision 'right'). Correct decisions cause transient increase in reward feedback R_t (red line). The numbers below the negative decision threshold label the segments. After a decision is taken, the current segment disappears and reward or penalty is given after a delay of 50ms. Stimuli were presented in random order. **b)** Top: performance as % correct in response to relevant stimuli steadily improves with learning and converges to a value close to optimal in 3000 trials (asymptotic *overall* performance was only slightly worse; not shown). Bottom: decision times for two stimuli vs. number of presentations of those stimuli. In both panels, curves were smoothed out with a low pass filter $\bar{x}_n = (1-\lambda)\bar{x}_{n-1} + \lambda x_n$, with $\lambda = 0.05$. **c)** Detail of decision times (top) and performance (bottom) for all 6 stimuli used in the task. In the top panel, the squares represent the total durations of the stimuli, dots are the sampled decision times in the last 100 trials, and crosses are the average decision times. After learning, the fastest decisions were in response to relevant segments, whereas decision times were fewer and closer to the maximal stimulus duration ($\sim$ 500ms) for non-relevant segments (key: **L**='go left', **R**='go right' and **N**='non-relevant stimuli')

became more confident about a decision, its response to the related stimulus became faster (best stimulus), whereas when stimuli had not been yet correctly identified, the decision times tended to be flat or increase during learning to allow for more information to be accrued (worst stimulus). In panel c), mean decision times and performance are shown for all stimuli (stimuli marked **N** were non-relevant stimuli). The best stimulus (panel b) was stimulus 5, for which performance reached 100% correct after training; the worst stimulus was stimulus 4, a non-relevant segment (like segment 3 in panel a). Note that the end-performance with this stimulus after training was $\sim 75\%$ correct (see panel c), bottom), which means that $\sim 25\%$ of the time the agent took an action during the presentation of this stimulus (the agent, however, is still learning to ignore this stimulus, see panel b), bottom, dashed line). Note that in the case of stimuli 1 and 2 the agent had become very confident of the correct decision, as implied by the short reaction times and the population activity overshoots above the decision threshold in the time interval between the decision and the rewarding feedback.

Simulation results with firing rate patterns (task 2). The previous scenario assumed that patterns of spike trains are reproduced exactly unmodified at each stimulus presentation ('frozen' spike patterns). This is clearly only a convenient starting point. A more realistic scenario could be based on firing rate coding, whereby a stimulus is defined by the firing rates of its input spike trains, collectively, but the spike times are generated anew during each stimulus presentation. This is both a more realistic scenario and a more challenging learning task for our spike-based learning rule. The stimuli were patterns of 60 Poisson spike trains with constant firing rate, each randomly sampled from values $6, 22, 40, 60$ spikes/s. The pattern of firing rates defining a stimulus were always fixed, but the actual spike times were generated anew at each stimulus presentation to produce Poisson spike trains with the given firing rates. By construction, all stimuli have the same overall firing rate, i.e., the stimuli could not be distinguished by unsupervised processing based on the overall firing rate of the input. The simulation shown in Fig. 3 confirms that the agent is able to identify relevant stimuli coded as patterns of firing rates, despite using a learning rule not explicitly designed to learn firing rates.

Conclusions and discussion. Learning to abstract relevant information from the environment is a crucial component of decision making; yet, current models typically assume that the relevant inputs are known to the decision maker, and defined once and for all. Here, we put forward a spiking network model able to detect stimuli from the environment based on their behavioral relevance. Since the stimuli are presented in sequence in a continuous stream, with unknown starting and ending points, the task is akin to temporal stimulus segmentation, i.e., the task of discovering boundaries between successive stimuli. Segmentation tasks such as ours are typically solved by methods such as Hidden Markov

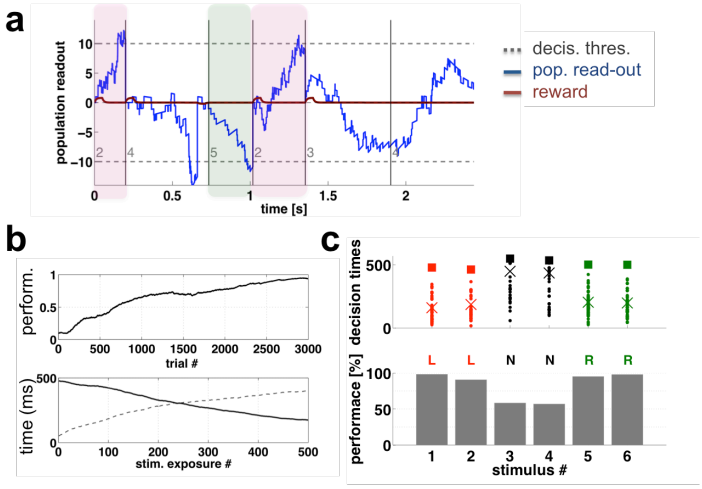


Figure 3: Simulation results with stationary firing rate patterns, same keys as Fig. 2. See the text for details.

Models [9], which require a-priori knowledge of the relevant stimuli (or at least the number of relevant stimuli), are not based on online algorithms, and lack biological plausibility. In contrast, our spiking network model learns online, does not require *a priori* knowledge of the relevant stimuli or even when they are being presented, it allows for direct comparison with neurobiological data and thus could help uncover potential correlates of decision confidence and other aspects of decision making. Another hallmark of our study is the use of 'information-controlled' tasks, which allows subjects to respond whenever they feel confident [8].

Our model differs from the class of neural-circuit models of decision making depicted in Fig. 1a, which require a neural population encoding each stimulus, and *a priori* knowledge of the relevant stimuli, and when they start and end. Following an approach more similar to ours, the 'tempotron' [10] can learn to separate spike patterns into two classes, which could be interpreted as 'relevant' vs. 'non-relevant'. However, the tempotron needs to know when stimuli start and end, and is given feedback for non-responses to relevant stimuli, which helps their identification. Also, the tempotron is only capable of binary decisions. Moreover, if applied in a population of neurons rather than in a single neuron, performance again slows down if no feedback from the population activity (resulting in an individual reward signal) is given.

This work can be extended in a number of directions. One could consider a visual segmentation task wherein a sequence of images slowly appear and disappear on top of a noisy background, and the task of the agent is to identify the images that are action-relevant. Preliminary simulations with a simple version of this task show encouraging results. A second direction is to go beyond 2 choice tasks. This could be obtained by subdividing the decision neurons into as many subpopulations as alternative decisions, with each subpopulation encoding a different decision. Each subpopulation would obey the same learning rule, which is aesthetically appealing and biologically plausible. Preliminary simulations show that with this modified architecture, the network also does a better job at learning to ignore non-relevant stimuli.

References

[1] X.-J. Wang. Decision making in recurrent neuronal circuits. *Neuron*, 60(2):215–34, Oct 2008.

[2] X.-J. Wang. Probabilistic decision making by slow reverberation in cortical circuits. *Neuron*, 36:955–968, 2002.

[3] S. Fusi, W. F. Asaad, E. K. Miller, and X.-J. Wang. A neural circuit model of flexible sensorimotor mapping: learning and forgetting on multiple timescales. *Neuron*, 54:319–333, Apr 2007.

[4] J.-P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Comput*, 18(6):1318–48, Jun 2006.

[5] R. Urbanczik and W. Senn. Reinforcement learning in populations of spiking neurons. *Nat Neurosci*, 12(3):250–2, Mar 2009.

[6] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(229), 1992.

[7] M. C. Wiener and B. J. Richmond. Model based decoding of spike trains. *Biosystems*, 67(1-3):295–300, 2002.

[8] J. Zhang, R. Bogacz, and P. Holmes. A comparison of bounded diffusion models for choice in time controlled tasks. *J Math Psychol*, 53(4):231–241, Aug 2009.

[9] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[10] R. Gütig and H. Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nat Neurosci*, 9(3):420–8, Mar 2006.

Scalable Bayesian Reinforcement Learning for Multiagent POMDPs

Christopher Amato
CSAIL
MIT
Cambridge, MA 02139
camato@csail.mit.edu

Frans A. Oliehoek
Department of Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
frans.oliehoek@maastrichtuniversity.nl

Eric Shyu
CSAIL
MIT
Cambridge, MA 02139
eshyu@mit.edu

Abstract

Bayesian methods for reinforcement learning (RL) allow model uncertainty to be considered explicitly and offer a principled way of dealing with the exploration/exploitation tradeoff. However, for multiagent systems there have been few such approaches, and none of them apply to problems with state uncertainty. In this paper, we fill this gap by proposing a Bayesian RL framework for multiagent partially observable Markov decision processes that is able to take advantage of structure present in many problems. In this framework, a team of agents operates in a centralized fashion, but has uncertainty about the model of the environment. Fitting many real-world situations, we consider the case where agents learn the appropriate models while acting in an online fashion. Because it can quickly become intractable to choose the optimal action in naïve versions of this online learning problem, we propose a more scalable approach based on sample-based search and factored value functions for the set of agents. Experimental results show that we are able to provide high quality solutions to large problems even with a large amount of initial model uncertainty.

Keywords: Multiagent Learning, Bayesian Reinforcement Learning, POMDPs

Acknowledgements

Research supported in part by AFOSR MURI project #FA9550-091-0538.

1 Introduction

Bayesian reinforcement learning (RL) techniques are promising in that, in principle, they provide an optimal exploration/exploitation trade-off with respect to the prior belief. In the context of multiagent systems, Bayesian RL has been used in stochastic games [2] and factored Markov decision processes (MDPs) [18]. These approaches assume the state of the problem is fully observable (or can be decomposed into fully observable components). Unfortunately, no approaches have been proposed that can model and solve problems with partial observability. In fact, while planning in partially observable multiagent domains has had some success (e.g., [1, 11]), very few multiagent RL approaches of any kind consider partially observable domains (notable exceptions, e.g., [3, 13]).

We propose a framework for Bayesian learning in multiagent systems with state uncertainty using multiagent partially observable Markov decision processes (MPOMDPs) that can exploit the multiagent structure in these problems. MPOMDPs represent a centralized perspective where all agents share the same partially observable view of the world and can coordinate on their actions, but have uncertainty about the underlying environment. To model this problem, we extend the Bayes-Adaptive POMDP (BA-POMDP) [15]—which represents beliefs over possible model parameters using Dirichlet distributions—to the multiagent setting. The resulting framework can be used as a Bayesian online learning approach which represents the initial model using priors and updates probability distributions over possible models as the agent acts in the real world. In particular, we utilize sample-based planning based on Monte Carlo tree search (MCTS) which has shown promise performing planning in large POMDPs [16] and Bayesian learning in large MDPs [6].

Unfortunately, these methods become ineffective as the number of (joint) actions and observations scales exponentially in the number of agents. To combat this intractability, we propose exploiting structure in the value functions associated with the agents. That is, many multiagent problems possess structure in the form of locality of interaction: agents interact directly with a subset of other agents. This structure enables a decomposition of the value function into a set of overlapping factors, which can be used to produce high quality solutions [5, 9, 10]. We propose two techniques for incorporating such factored value functions into MCTS, thereby mitigating the additional challenges for scalability imposed by the exponential number of joint actions and observations. This approach is the first MCTS variant to exploit structure in multiagent systems, achieving better sample complexity and improving value function generalization by factorization.

2 Background

MPOMDPs form a framework for multiagent planning under uncertainty for a team of agents. At every stage, agents take individual actions and receive individual observations. However, in an MPOMDP, the assumption is that the team of agents is acting in a 'centralized manner', which means that we assume that all individual observations are shared via communication. We will restrict ourselves to the setting where such communication is free of noise, costs and delays.

Formally, an MPOMDP is a tuple $\langle I, S, \{A_i\}, T, R, \{Z_i\}, O, h\rangle$ with: I, a finite set of agents; S, a finite set of states with designated initial state distribution b_0; $A = \times_i A_i$, the set of joint actions, using action sets for each agent, i; T, a set of state transition probabilities: $T^{s\vec{a}s'} = \Pr(s'|s,\vec{a})$, the probability of transitioning from state s to s' when the set of actions $\vec{a}$ are taken by the agents; R, a reward function: $R(s,\vec{a})$, the immediate reward for being in state s and taking the set of actions $\vec{a}$; $Z = \times_i Z_i$, the set of joint observations, using observation sets for each agent, i; O, a set of observation probabilities: $O^{\vec{a}s'\vec{z}} = \Pr(\vec{z}|\vec{a},s')$, the probability of seeing the set of observations $\vec{o}$ given the set of actions $\vec{a}$ was taken which results in state s'; h, the number of steps before termination or horizon. An MPOMDP can be reduced to a special type of POMDP in which there is a single centralized controller that takes joint actions and receives joint observations [14].

Most research concerning POMDPs has considered the task of *planning*: given a full specification of the model, determine an optimal (joint) policy, π, mapping past (joint) observation histories (which can be summarized by distributions $b(s)$ over states called beliefs) to (joint) actions. Such an optimal (joint) policy can be extracted from an optimal Q-value function, $Q(b,a) = \sum_s R(s,a) + \sum_z P(z|b,a) \max_{a'} Q(b',a')$, by acting greedily, in a way similar to the situations in regular MDPs [17]. Computing $Q(b,a)$, however, is complicated by the fact that the space of beliefs is continuous [7].

While POMDP planning methods can find solutions effectively given a problem model, for many real-world applications, the model is not (perfectly) known in advance, requiring the agents to learn about their environment during execution. To deal with such partially observable multiagent learning problems, we build on the framework of Bayes-Adaptive POMDPs [15]. This approach utilizes Dirichlet distributions to model uncertainty over both transitions and observations.

Intuitively, if the agent could observe both states and observations, it could maintain vectors ϕ and ψ of counts for transitions and observations respectively. That is, $\phi^a_{ss'}$ is the transition count representing the number times state s' resulted from taking action a in state s and $\psi^a_{s'z}$ is the observation count representing the number of times observation z was seen after taking action a and transitioning to state s'. While the agent cannot observe the states and has uncertainty about the actual count vectors, *this uncertainty can be represented using the regular POMDP formalism*. That is, the count vectors are included as part of the hidden state of a special POMDP, called BA-POMDP.

3 BA-MPOMDPs

The BA-POMDP can be extended to the multiagent setting in a straightforward manner as the Bayes-Adaptive multiagent POMDP (BA-MPOMDP). The BA-MPOMDP model allows a team of agents to learn about its environment while acting in a Bayesian fashion and is applicable in any multiagent RL setting where there is instantaneous communication. Since a BA-MPOMDP can be simply seen as a BA-POMDP where the actions are joint actions and the observations are joint observations, the theoretical results related to BA-POMDPs also apply to the BA-MPOMDP model.

Formally, a BA-MPOMDP is a tuple $\langle I, S_{BM}, \{A_i\}, T_{BM}, R_{BM}, \{Z_i\}, O_{BM}, h\rangle$ where $I, \{A_i\}, \{Z_i\}, h$ are as before. The state of the BA-MPOMDP now includes the Dirichlet parameters (i.e., the count vectors): $s_{BM} = \langle s, \phi, \psi\rangle$. As such, the set of states is given by $S_{BM} = S \times \mathcal{T} \times \mathcal{O}$ where $\mathcal{T} = \{\phi \in \mathbb{N}^{|S||A||S|} | \forall (s, \vec{a}) \sum_{s'} \phi^{\vec{a}}_{ss'} > 0\}$ is the space of all possible transition counts and similarly $\mathcal{O}$ is the space of all possible observation parameters: $\mathcal{O} = \{\psi \in \mathbb{N}^{|S||A||Z|} | \forall (s,\vec{a}) \sum_{\vec{z}} \psi^{\vec{a}}_{s'\vec{z}} > 0\}$ where $|A|$ is the number of joint actions and $|Z|$ is the number of joint observations.

In order to define T_{BM}, O_{BM}, the transition and observation probabilities for the BA-MPOMDP, we need the expected transition and observation probabilities induced by (the count vectors of) a state: $T^{s\vec{a}s'}_{\phi} = \mathbf{E}[T^{s\vec{a}s'}|\phi] = \phi^{\vec{a}}_{ss'}/N^{s\vec{a}}_{\phi}$, $O^{\vec{a}s'\vec{z}}_{\psi} = \mathbf{E}[O^{\vec{a}s'\vec{z}}|\psi] = \psi^{\vec{a}}_{s''\vec{z}}/N^{\vec{a}s'}_{\psi}$, where $N^{sa}_{\phi} = \sum_{s''} \phi^{a}_{ss''}$, and $N^{as'}_{\psi} = \sum_{z'} \psi^{a}_{s'z'}$. The transition probabilities $P(\langle s', \phi', \psi'\rangle | \langle s, \phi, \psi\rangle, a)$ can be defined using a vector $\delta^{a}_{ss'}$ which is 1 at the index of a, s and s' and 0 otherwise: $T_{BM}((s,\phi,\psi),\vec{a},(s',\phi',\psi')) = T^{s\vec{a}s'}_{\phi} O^{\vec{a}s'z}_{\psi}$ if $\phi' = \phi + \delta^{\vec{a}}_{ss'}$ and $\psi' = \psi + \delta^{\vec{a}}_{s'z}$(and 0 otherwise). Similarly, for observations, we define $\delta^{\vec{a}}_{s'\vec{z}}$ to be a vector with value 1 at the index $\vec{a}$, s' and $\vec{z}$ and 0 otherwise: $O_{BM}((s,\phi,\psi),\vec{a},(s',\phi',\psi'),z) = 1$ if $\phi' = \phi + \delta^{\vec{a}}_{ss'}$ and $\psi' = \psi + \delta^{\vec{a}}_{s'z}$(and 0 otherwise). The reward model remains the same (since it is assumed to be known), $R_{BM}((s,\phi,\psi),\vec{a}) = R(s,\vec{a})$. We assume the initial state distribution b_0 and initial count vectors ϕ_0 and ψ_0 are given.

4 Monte Carlo Tree Search for Multiagent POMDPs

Monte Carlo Tree Search for POMDPs A successful recent online planning method, called partially observable Monte Carlo planning (POMCP) [16], extends Monte Carlo tree search (MCTS), and in particular the UCT algorithm [8], to solving POMDPs. At every stage, the algorithm performs online planning, given the current belief, by incrementally building a lookahead tree that contains (statistics that represent the) $Q(b,a)$. The algorithm, however, avoids expensive belief updates by creating nodes not for each belief, but simply for each action-observation history h. In particular, it samples hidden states s only at the root node (called 'root sampling') and uses that state to sample a trajectory that first traverses the lookahead tree and then performs a (random) rollout. The return of this trajectory is used to update the statistics for all visited nodes. When traversing the tree, actions are selected to maximize the 'upper confidence bounds': $U(h,a) = Q(h,a) + c\sqrt{\log(N+1)/n}$. Here, N is the number of times the history has been reached and n is the number of times that action a has been taken in that history. When the exploration constant c is set correctly, POMCP can be shown to converge in the limit. Moreover, the method has demonstrated good performance in large domains with a limited numbers of simulations.

Because the BA-MPOMDP formalism constructs an (infinite state[1]) POMDP, POMCP could be applied here too. Doing so means that during online planning, a lookahead tree will be constructed that has nodes corresponding to *joint* action observation histories $\vec{h}$, and where statistics are stored that represent the expected values $Q(\vec{h},\vec{a})$ and upper confidence bounds $U(\vec{h},\vec{a})$. A shortcoming of Monte Carlo tree search methods is that they are not directly suitable for multiagent problems due to the large number joint actions and joint observations, which are exponential in the number of agents.

The large number of joint observations is problematic, since it will lead to a lookahead tree with very high branching factor and a breakdown of particle filtering to estimate the belief (necessitating starting from the initial belief again, or acting using a separate policy such as a random one). The large number of actions results in exponentially many joint actions that have to be selected at least a few times to drive down their confidence bounds (i.e., exploration bonus).

Coordination Graphs In many cases, the effect of a joint action is factorizable as the effects of the action of individual agents or small groups of agents. For instance, consider a team of agents that is tasked with fighting fire at a number of burning houses, as illustrated in Fig. 1(a). In such a setting, the overall transition probabilities can be factored as a product of transition probabilities for each house [12], and the transitions of a particular house may depend only on the amount of water deposited on that house (rather than the exact joint action).

We can consider agents' interactions in the form of a coordination graph which represents interactions between subsets of agents and permits factored linear value functions as an approximation to the joint value function. Specifically—for the moment assuming a stateless problem—an action-value function can be approximated by $Q(\vec{a}) = \sum_e Q_e(\vec{a}_e)$, where each component e is a value specified over only a (possibly overlapping) subset of agents. Note that in this and later

[1]Since there can be infinitely many count vectors, the state space is infinite, but a finite approximate model can be used [15].

(a) (b)

Figure 1: (a) Illustration of an MPOMDP in which a team of agents has to fight a fire, and (b) Illustration of the coordination graph (as a factor graph) with houses represented as h1,...,h4 and agents represented as a1,...,a3. Each agent should coordinate with the adjacent agents in the graph.

formulations, a normalization term of $1/|e|$ can be used to scale the Q-values back to the range of the original problem, but the maximizing actions remain the same.

In cases where such a factorization holds, the maximization $\max_{\vec{a}} \sum_e Q_e(\vec{a}_e)$ can be performed efficiently via variable elimination (VE) [5], or max-sum [4, 9]. These algorithms are not exponential in the number of agents (although VE is exponential in the induced with), and therefore enable significant speed-ups for larger number of agents.

Factored Statistics The first technique we introduce, called *Factored Statistics* directly applies the idea of coordination graphs inside MCTS. Rather than maintaining one set of statistics in each node that expresses the expected value for each joint action $Q(\vec{h}, \vec{a})$, we maintain several sets of statistics, each one expressing the value for a set of agents $Q_e(\vec{h}, \vec{a}_e)$. As such, the Q-value function is approximated by $Q(\vec{h}, \vec{a}) \approx \sum_e Q_e(\vec{h}, \vec{a}_e)$.

Since this method retains fewer statistics and performs joint action selection more efficiently via VE, we expect that this approach will be more efficient than plain application of POMCP to the BA-MPOMDP. However, the complexity due to joint observations is not directly addressed: because joint histories are used, reuse of nodes and the ability to create nodes in the tree for the necessary observations seen during execution may be limited.

Factored Trees The second technique, called *Factored Trees*, additionally tries to overcome this burden of the large number of joint observations. This is done by further decomposing the joint histories into local histories over factors. That is, in this case, the Q-values are approximated by $Q(\vec{h}, \vec{a}) \approx \sum_e Q_e(\vec{h}_e, \vec{a}_e)$. This approach further reduces the number of statistics maintained and increases the reuse of nodes in MCTS and the chance that nodes in the trees will exist for observations that are seen during execution. As such, it aims to increase performance by utilizing more generalization (now also over local histories), as well as producing more robust particle filters.

Finally, we note that this type of factorization has major implications for the implementation of the approach: rather than constructing a single tree, we now need to construct a number of trees in parallel, one for each factor (or edge in the coordination graph) e. A node of the tree of an component e now stores the required statistics: $N_{\vec{h}_e}$, the count for the local history, $n_{\vec{a}_e}$, the counts for actions taken in the local tree and Q_e for the tree.

5 Experimental Results

We performed an evaluation on the firefighting problem from Section 4. Each experiment was run for a given number of simulations (the number of samples used at each step to choose an action) and averaged over a number of runs (resetting the state and count vectors to their initial values). To determine the value of acting with the true model known, we provide results from POMCP [16] and to show the result of acting solely based on the initial prior given by the initial count vectors, we provide results from a 'No learning' method. This no learning approach uses the BA-MPOMDP in the same way as the other methods, but never updates the count vectors, causing it to retain the same uncertain distribution over models. We also provide results for the value produced by uniform random action selection. The values given are the average undiscounted returns for the horizon (i.e., number of steps in the problem) shown. Experiments were run on a single core of a 2.5 GHz machine with 8GB of memory.

The fire fighting domain [12] consists of four agents and five houses, each with 3 different fire levels. Fires are suppressed more quickly if a larger number of agents choose that particular house. Fires also spread to neighbor's houses and can start at any house with a small probability. Priors were used that had high confidence in (near) correct transition probabilities and low confidence in incorrect (near uniform) observation probabilities.

Results are shown in Table 1 where the benefits of the factored approaches are seen. For a small number of samples (which is crucial on large problems, $|S| = 243$, $|A| = 81$, $|Z| = 16$) the factored tree method learns very quickly, providing significantly better values. Using factored statistics will learn more slowly, but the value function is closer to optimal due to the use of the full history. BA-MPOMDP and No learning perform poorly due to the incorrect prior and insufficient

	Horizon 10	
	50 Simulations	250 Simulations
POMCP (true)	-87.3 ± 2.09	-50.6 ± 9.50
Factored statistics	-47.1 ± 7.85	-21.8 ± 1.94
Factored tree	-41.2 ± 3.85	-29.8 ± 5.64
BA-MPOMDP	-85.6 ± 1.66	-51.6 ± 5.94
No learning	-86.7 ± 1.59	-54.6 ± 6.41
Random	-81.6 ± 4.09	-81.6 ± 4.09

	Horizon 50
	50 Simulations
POMCP (true)	-425.7 ± 4.65
Factored statistics	-403.9 ± 10.78
Factored tree	-210.2 ± 12.01
BA-MPOMDP	-436.9 ± 4.66
No learning	-436.9 ± 4.66
Random	-437.6 ± 5.73

Table 1: Undiscounted return (and standard error) for horizon 10 and 50 fire fighting problems averaged over 10 runs

samples to choose high-quality actions. After more samples (as seen by 250 samples), the performance of the flat models improve, but the factored methods still perform better. POMCP(true) with 100000 simulations was able to achieve values for horizon 10 of -19.83 ± 0.96 and 50 of -62.1 ± 6.96. Note that the 'No learning' method gives the value of using the prior without learning, showing the benefit of the learning approaches. In particular, the factored approaches are able to improve learning through generalization and make better use of the statistics and the particle filter.

6 Conclusions

We present the first method to utilize multiagent structure to produce a scalable method for multiagent Bayesian reinforcement learning with state uncertainty. To combat exponential growth of the number of joint actions and observations, we propose two methods for decomposing the agents using a coordination graph to reduce 1) the number of joint actions and 2) the number of joint histories considered. These methods are used in conjunction with a leading POMDP method, POMCP [16], to generate a MCTS-based sample-based planner for our Bayes-Adaptive MPOMDP model. Our experimental results demonstrate that the proposed techniques allow agents to both learn faster (with fewer simulations) and produce higher quality solutions. We expect that these approaches can serve as the basis for many future work directions in multiagent learning as well as be used to solve BA-POMDPs with large action and observation spaces.

References

[1] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer. Decentralized control of partially observable Markov decision processes. In *CDC*, 2013.

[2] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *AAMAS*, 2003.

[3] Y.-H. Chang, T. Ho, and L. P. Kaelbling. All learning is local: Multi-agent learning in global reward games. In *NIPS 16*, 2004.

[4] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS*, 2008.

[5] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *NIPS*, 15, 2001.

[6] A. Guez, D. Silver, and P. Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *NIPS 12*, 2012.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *AIJ*, 101, 1998.

[8] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *ECML*, 2006.

[9] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *JMLR*, 7, 2006.

[10] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: a synthesis of distributed constraint optimization and POMDPs. In *AAAI*, 2005.

[11] F. A. Oliehoek. Decentralized POMDPs. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State of the Art*. Springer, 2012.

[12] F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *AAMAS*, 2008.

[13] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *UAI*, pages 489–496, 2000.

[14] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 16, 2002.

[15] S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. A Bayesian approach for learning and planning in partially observable Markov decision processes. *JAIR*, 12, 2011.

[16] D. Silver and J. Veness. Monte-carlo planning in large POMDPs. In *NIPS 23*, 2010.

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

[18] W. T. L. Teacy, G. Chalkiadakis, A. Farinelli, A. Rogers, N. R. Jennings, S. McClean, and G. Parr. Decentralized Bayesian reinforcement learning for online agent collaboration. In *AAMAS*, 2012.

Communicating with Unknown Teammates

Samuel Barrett
Dept. of Computer Science
The Univ. of Texas at Austin
Austin, TX 78712 USA
sbarrett@cs.utexas.edu

Noa Agmon
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, 5290002 Israel
agmon@macs.biu.ac.il

Noam Hazon
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, 5290002 Israel
hazonn@macs.biu.ac.il

Sarit Kraus[1,2]
[1]Dept. of Computer Science
Bar-Ilan University
Ramat Gan, 5290002 Israel
[2]Inst. for Advanced Computer Studies
University of Maryland
College Park MD 20742
sarit@macs.biu.ac.il

Peter Stone
Dept. of Computer Science
The Univ. of Texas at Austin
Austin, TX 78712 USA
pstone@cs.utexas.edu

Abstract

Teamwork is central to many tasks, and past research has introduced a number of methods for coordinating teams of agents. However, with the growing number of sources of agents, it is likely that an agent will encounter teammates that do not share its coordination method. Therefore, it is desirable for agents to adapt to these teammates, forming an effective *ad hoc team*. Past ad hoc teamwork research has focused on cases where the agents do not directly communicate. This paper tackles the problem of communication in ad hoc teams, introducing a minimal version of the multiagent, multi-armed bandit problem with limited communication between the agents. The theoretical results in this paper prove that this problem setting can be solved in polynomial time when the agent knows the set of possible teammates. Furthermore, the empirical results show that an agent can cooperate with a variety of teammates not created by the authors even when its models of these teammates are imperfect.

Keywords: Ad Hoc Teams, Multiagent Systems, Teamwork, Multi-armed Bandits

1 Introduction

Given the growing number of both software and robotic agents, effective teamwork is becoming vital to many tasks. Robots are becoming cheaper and more durable, and software agents are becoming more common for tasks including bidding in ad auctions. These agents are being developed by an increasing number of companies and research laboratories. As the number of sources of agents grows, so does the need for agents to cooperate with a variety of different teammates.

This need is addressed in the area of *ad hoc teamwork*, where agents are evaluated in their ability to cooperate with a variety of teammates. Stone et al. [12] define ad hoc teamwork problems as problems in which a team cannot pre-coordinate its actions, and they argue that evaluating an ad hoc team agent fundamentally depends on both the domains it may face as well as the teammates it can encounter. To this end, they introduce an evaluation algorithm that includes this consideration.

Past work on ad hoc teamwork has focused on the case where the ad hoc agent cannot directly communicate to its teammates. Instead, the focus of this work is on how an agent can influence its teammates through limited communication when a common language exists and the agent has more knowledge than its teammates. However, the ad hoc agent cannot influence how its messages are interpreted, only the messages it sends. This work has three main contributions, the first being the introduction of a minimal domain for investigating teammate communication. The second contribution is proving that one scenario is solvable in polynomial time. However, for practical use, the polynomial algorithm does not scale well, so the third contribution is the evaluation of an empirical planning algorithm in this domain.

The work in this paper shows that ad hoc agents can optimally learn about their environment and their teammates while acting and communicating. This learning is tractable and can be performed in polynomial time in terms of the problem parameters. In addition, even when it has imperfect assumptions about its teammates, an ad hoc agent can still learn and adapt so as to enable its team to perform effectively.

2 Problem Description

This paper focuses on a multiagent, multi-armed bandit problem that allows limited communication because it serves as a minimal decision making domain that exhibits the necessary properties for investigating communication with unknown teammates. In ad hoc teamwork, the goal is to create agents that can cooperate with a variety of possible teammates. We assume that a number of agents are pre-designed to cooperate to achieve the given task, and we want to design an agent that can fit into this team. Matching the behavior of the other agents is either infeasible due to not knowing their behaviors or undesirable if we have access to additional knowledge or better algorithms to give our agent.

Formally, the bandit problem in this paper is given by the tuple $G = (\mathbb{A}, \mathbb{C}, \mathbb{P}, T)$ where $\mathbb{A}$ is a set of two arms $\{arm_1, arm_2\}$ with Bernoulli payoff distributions, returning either 0 or 1, $\mathbb{C}$ is a set of possible communications to send and their costs, $\mathbb{P}$ denotes the players in the problem with $|\mathbb{P}| = n + 1$, and T is the number of rounds. Each round has a communication phase followed by an action phase, and, in both, all agents act simultaneously. In the communication phase, each agent can broadcast a message of each of the following types or send no message, each with an associated cost ($cost(m)$):

- **obs** – Send the agent's last selected arm and payoff
- **mean**$_{arm}$ – Send the agent's observed mean and number of pulls for the specified arm
- **suggest**$_{arm}$ – Suggest that the agent's teammates pull the specified arm

In the action phase, each agent chooses a single arm and observes a payoff from that arm. The team's goal is to maximize the total rewards from the arms minus the costs of communication. Since the ad hoc agent's teammates form an existing team, we assume that they are tightly coordinated, i.e. that the team's behavior can be described as a function of the team's total number of pulls and successes of each arm, but only the ad hoc agent's pulls and successes that it has communicated.

3 Preliminaries

Before discussing the theoretical analysis applied to this problem, it is important to understand the theoretical models that will be used in our analysis. In Section 4, we model this problem as a Markov Decision Process (MDP). An MDP is a 4-tuple $M = (S, A, P, R)$ where S is a set of states, $A(s)$ is the set of actions available from state $s \in S$, $P(s, a, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the transition function specifying the probability of reaching state s' after taking action a in state s, and $R(s, a, s')$ is the resulting immediate reward. In an MDP, the goal is to find an optimal policy $\pi^*(s)$ that selects actions in order to maximize the long term expected reward received. An extended version of this model known as the Partially Observable Markov Decision Process (POMDP) is used when an agent receives observations, $\Omega(s) = o \in O$, rather than the true state. The underlying states and transition function remain unchanged from the original MDP, but the agent now has a harder task, as it must additionally reason about which underlying state it is in.

When reasoning about the difficulty of solving a POMDP, one approach is to reason about the δ-covering of its belief space. For a metric space A, a set B is a *δ-covering* if $\forall a \in A \ \exists b \in B$ such that $|a - b| < \delta$. Intuitively, a δ-covering can be thought of as a set of multi-dimensional balls filling a space. The covering number refers to the number of δ-balls that are required to cover the belief space. From Theorem 1 in [9], it is known that a POMDP can be approximately solved in time polynomial in terms of the size of its covering number. While this theorem shows this result for the infinite horizon, discounted rewards case, these results extend to the finite horizon setting.

The difficulty of solving MDPs and POMDPs depends on the size of the state and action spaces, so we define the size of these spaces here. The behavior of the teammates depends on the teams' observations of the arms as well as the messages the ad hoc agent has sent. Inspecting the number of possible outcomes shows that there are at most $6n^4T^{13}$ states. In the communication phase, the ad hoc agent can optionally send a message of each type, result in 18 possible actions.

4 Theoretical Analysis

We investigate the version of the bandit problem where the ad hoc agent knows that its teammates' behaviors are drawn from a continuously parameterized set of stochastic behaviors. We consider a small number of possible behaviors, specifically ε-greedy and UCB(c). For these behaviors, ε is the probability of taking a random action, and c is the scaling factor of the confidence bound. Note that while we only use two models for simplicity, this analysis can be extended for any fixed number of models.

To tackle this problem, we model it as a POMDP with three partially observed values: ε, c, and the probability of the teammates being ε-greedy versus UCB(c). The transition function for the fully observable state variables remains the same as the original MDP. The probabilities of the two models are updated given the probability that each of the models would have predicted the observed actions, and the updates to the probability distributions of ε and c are described in Lemma 1. The remainder of the POMDP remains as defined above.

In Lemma 1 and Theorem 2, we show that in this version of the problem, the ad hoc agent can perform within η of the optimal behavior with calculations performed in polynomial time. This result comes from reasoning about the δ-covering of the belief space, which defines the difficulty of solving the POMDP as discussed in Section 3.

Lemma 1. *The belief space of the resulting POMDP has a δ-covering with size poly$(T, n, 1/\delta)$.*

Proof. Using Proposition 1 of [7], we know that the fully observed state variables result in a multiplicative factor that is polynomial in T and n. Therefore, we focus our analysis on the remaining unobserved variables. The belief space over the probability between the two models is a single real value in [0,1], resulting in a factor of $1/\delta$. The parameter ε has a uniform prior, so the posterior is a beta distribution, relying on two parameters, α and β. These parameters correspond to the (fully observed) number of observed greedy and random pulls; thus, each are integers bounded by nT. Therefore, the probability distribution over ε can be represented using a factor of size $(nT)^2$.

The parameter c has a uniform prior, and UCB agents choose based on comparing $\frac{s_i+s_i^c}{p_i+p_i^c} + c\sqrt{\frac{ln(p_0+p_0^c+p_1+p_1^c)}{p_i+p_i^c}}$ for $i = 1, 2$. Using linear programming, pieces of the range of c can be eliminated by observing the actions of the teammates. However, the posterior remains uniform; only the range changes. Given the nature of c, the eliminated pieces of the range must be at the top or bottom of the current range of c. Therefore, the probability distribution over c can be represented using two real values in $[0, 1]$ that are the top and bottom of the uniform range of c, resulting in a factor of $1/\delta^2$. Combining these all of these factors results in a δ-covering of size poly$(T, n, 1/\delta)$. □

As discussed in Section 3, a POMDP can be solved approximately in polynomial time in terms of the size of its covering number. Given this result and Lemma 1, Theorem 2 follows directly.

Theorem 2. *If an ad hoc agent observes its teammates' actions, knows the true arm distributions, and knows that its teammates are drawn from a continuous set of ε-greedy and UCB teammates, it can calculate an η-optimal behavior in poly$(n, T, b, 1/\eta)$ time.*

5 Empirical Evaluation

While the previous section focused on proving that our formulations of the multi-armed bandit problem can be solved in polynomial time, the existing techniques for calculating exact solutions are impractical for solving problems with more than a handful of rounds and more than two arms. Therefore, in the empirical setting, we use Partially Observable Monte-Carlo Planning (POMCP) [11], which has been shown to be effective on a number of large POMDPs.

5.1 Methods

POMCP is a Monte Carlo Tree Search (MCTS) algorithm that is based on the Upper Confidence bounds for Trees (UCT) algorithm [8]. Specifically, POMCP starts from the current state of the problem and performs a number of simulations until reaching the end of the problem. For its teammates, the ad hoc agent plans as if they are selecting actions using

either the ε-greedy or the UCB algorithms. To model the effects of suggestions, agents are given some probability of following the suggestion rather than taking their regular action, with the probability being uniformly drawn from [0,1] at the beginning of an episode. In all of the evaluations, we assume that the ad hoc agent can observe its teammates' actions and payoffs.

While we evaluate the ad hoc agent when it encounters teammates that are using the ε-greedy and the UCB algorithms, we also consider a number of agents that were not created by the authors, denoted *externally-created teammates*. These agents were designed by undergraduate and graduate students as part of an assignment on agent design. To prevent any bias in the creation of the agents, the students designed the entire team and were unaware of the ad hoc teamwork problem. These agents were given the same three types of messages available to the ad hoc agent. Note that these teammates are not tightly coordinated and their behavior does not match our models.

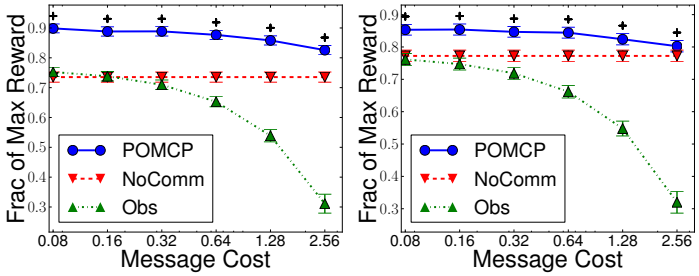
5.2 Results

All of the evaluations use 100 trials with randomly selected teams. In this analysis, the ad hoc agent initially samples a number of ε-greedy and UCB teams with random parameter values. The results are the average team rewards normalized by the average reward if every agent continuously pulled the best arm. Statistical significance is tested with a paired Student-T test with $p < 0.05$ and is denoted with a "+" in the figures when comparing POMCP to all other methods.

We compare three behaviors of the ad hoc agent:

- **NoComm** - Always pulls the best arm and does not communicate
- **Obs** - Always pulls the best arm and communicates its last observation
- **POMCP** - Plans using POMCP which arm to pull and what to communicate

NoComm and Obs serve as baselines. Unless otherwise specified, there are 3 arms, 10 rounds, and 7 externally-created teammates to test how our approach scales to bigger problems than are theoretically proven. Furthermore, the costs for sending messages are randomly selected for each run, and all agents are informed of the costs. To model the size of different messages and allow for varied communication scenarios, the cost of sending the last observation is selected from $[0, 2m]$ (arm and payoff), the cost of sending the mean of an arm is in $[0, 3m]$ (arm, pulls, and successes), and the cost of suggesting an arm is in $[0, m]$ (arm), where $m = 0.75$ unless specified.



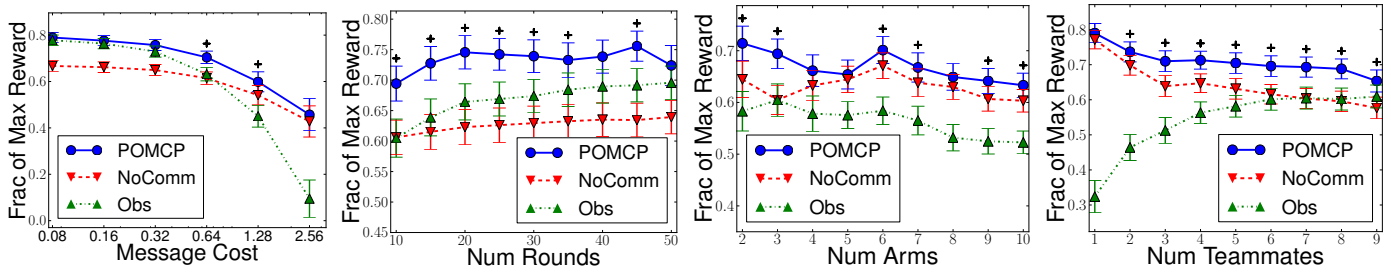
(a) ε-greedy teammates (b) UCB teammates

Figure 1: Normalized rewards with varied message costs with a logarithmic x-axis. Significance is denoted by "+"

Figure 1 presents the results when the ad hoc agent encounters the problem discussed in Section 4, cooperating with teams that are ε-greedy or UCB, with varied message costs. Note that NoComm is unaffected by the message costs as it does not communicate. The results indicate that the agent can effectively plan its actions, significantly outperforming the baselines. When the ad hoc agent knows the correct behavior type, the results are similar to knowing that either ε-greedy or UCB teams are possible.

On the other hand, Figure 2 shows the results with externally-created agents, a problem not covered by any theoretical guarantees, as the models do not match the true teammates. If all agents start with no observations of the arms, all of the considered behaviors for the ad hoc agent perform similarly because the teammates usually quickly converge to the best arm. Therefore, for these results, we consider the case where in the first 5 rounds, the teammates' pulls of the best arm are biased to have a lower chance of success. Then, we evaluate how well the ad hoc agents help correct their teammates' biases. In these evaluations, we test the sensitivity of the agent to various problem parameters. Note that the message costs are also applied to the externally-created teammates, which are informed of the current message costs, so the performance of NoComm is also affected by message costs.

As the cost of communicating increases, NoComm becomes closer to the optimal behavior. As the number of rounds increases, communicating is more helpful because there is more time to reap the benefits of better informing the teammates. With more arms, it is harder to get the teammates to select the best arm, so communicating is less helpful. With more teammates, communicating is more likely to be outweighed by other agents' messages, but there is more benefit if the team can be convinced, hence the improvement of Obs. Overall, the results in all of these scenarios tell a similar story, specifically that reasoning about communication helps an ad hoc agent effectively cooperate with various teammates, even when its models of these teammates are incomplete or incorrect.



(a) Message costs with logarithmic x-axis. (b) Numbers of rounds. (c) Numbers of arms. (d) Numbers of teammates.

Figure 2: Normalized rewards with varied parameters when cooperating with externally-created teammates.

6 Related Work

Bowling and McCracken [3] consider robots playing soccer in which the ad hoc agent has a playbook that differs from its teammates'. In [10], Liemhetcharat and Veloso reason about selecting agents to form ad hoc teams. Barrett et al. [2] empirically evaluate an MCTS-based ad hoc team agent in the pursuit domain, and Barrett and Stone [1] analyze existing research on ad hoc teams and propose one way to categorize ad hoc teamwork problems. A more theoretical approach is Wu et al.'s work [13] into ad hoc teams using stage games and biased adaptive play.

Ad hoc teamwork is also closely related to the area of opponent modeling, differing in whether one models teammates or opponents. Interacting with opponents often requires reasoning about worst case scenarios. One promising approach for opponent modeling is the AWESOME algorithm [4], which tackles repeated games and guarantees convergence and rationality. Further work investigates agents that explicitly model and reason about their opponent's beliefs in the form of interactive POMDPs [6] and interactive dynamic influence diagrams (I-DIDs) [5].

7 Conclusion

Past work into ad hoc teamwork has largely focused on scenarios in which the ad hoc agent cannot directly communicate with its teammates. This work addresses this gap by introducing a minimal domain with communication, where ad hoc agent controls what messages to send to its teammates, but it cannot control their reactions to the messages. Our analysis proves that ad hoc team agents can optimally cooperate in some scenarios using only polynomial computation. Furthermore, this paper evaluates an empirical algorithm for planning in ad hoc teamwork problems. This algorithm is shown to be effective when cooperating with teammates created by a variety of developers even when planning with imperfect models. These results show that an ad hoc agent can simultaneously learn about its teammates and the environment to enable its team to perform effectively.

References

[1] S. Barrett and P. Stone. An analysis framework for ad hoc teamwork tasks. In *AAMAS '12*, June 2012.

[2] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *AAMAS '11*, May 2011.

[3] M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, pages 53–58, 2005.

[4] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67, May 2007.

[5] P. Doshi and Y. Zeng. Improved approximation of interactive dynamic influence diagrams using discriminative model updates. In *AAMAS '09*, 2009.

[6] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *JAIR*, 24(1):49–79, July 2005.

[7] D. Hsu, W. S. Lee, and N. Rong. What makes some POMDP problems easy to approximate? In *Advances in Neural Information Processing System*. 2007.

[8] L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo planning. In *ECML '06*, 2006.

[9] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.

[10] S. Liemhetcharat and M. Veloso. Modeling mutual capabilities in heterogeneous teams for role assignment. In *IROS '11*, pages 3638 –3644, 2011.

[11] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *NIPS '10*. 2010.

[12] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI '10*, July 2010.

[13] F. Wu, S. Zilberstein, and X. Chen. Online planning for ad hoc autonomous agent teams. In *IJCAI*, 2011.

Online Learning in Markov Decision Processes with Changing Reward Sequences

Travis Dick, András György, and Csaba Szepesvári
Department of Computing Science, University of Alberta
Edmonton, AB, Canada
tdick,gyorgy,csaba.szepesvari@ualberta.ca

Abstract

In this paper we consider online learning in finite Markovian Decision Process with changing reward sequences under full and bandit-information. We propose to view this problem as an instance of online linear optimization. We propose two methods for this problem: MD^2 (mirror descent with approximate projections) and the continuous exponential weights algorithm with Dikin walks. We provide a rigorous complexity analysis of these techniques, while providing near-optimal regret-bounds. In the case of full-information feedback, our results complement existing results, while in the case of bandit-information feedback, we manage to improve the dependence of regret significantly by removing the restrictive assumption that the state-visitation probabilities are uniformly bounded away from zero under all policies.

1 Introduction

We consider the problem of online learning in discrete time finite Markov decision processes (MDPs) with arbitrarily changing cost processes. It is assumed that a learner moves in a finite state space $\mathcal{X}$. Occupying a state x_t at time instant t, the learner takes an action $a_t \in \mathcal{A}(x_t)$, where $\mathcal{A}(x_t)$ is some finite set of actions available at state x_t. Then the agent moves to some new random state x_{t+1}, where the distribution of x_{t+1}, given x_t and a_t is determined by the Markov transition kernel $P(\cdot|x_t, a_t)$. Simultaneously, the agent receives some immediate cost $f_t(x_t, a_t)$, where the cost function $f_t : \mathcal{U} \to [0, 1]$ is assumed to be bounded and $\mathcal{U} = \{(x, a) : x \in \mathcal{X}, a \in \mathcal{A}(x)\}$. The goal of the learner is to minimize its total cost. We assume here that the cost function f_t can change in an arbitrary manner between time instants. The performance of the learner is measured against the best stationary policy in hindsight, giving rise to the expected regret:

$$\mathbf{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t, a_t)\right] - \min_{\pi} \mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t^{\pi}, a_t^{\pi})\right].$$

Here, for a given stationary policy π (*i.e.* , π is such that $\pi(\cdot|x)$ is a probability distribution over $\mathcal{A}(x)$ for any $x \in X$), (x_t^{π}, a_t^{π}) denotes the state-action pair that policy π would visit in time step t if this policy was was used from $t = 1$ (we may assume that $x_1^{\pi} = x_1$). Note that a sublinear regret-growth, $\mathbf{R}_T = o(T)$ ($T \to \infty$) means that the *average* cost collected by the learning agent approaches that of the best policy in hindsight. Naturally, a smaller growth-rate is more desirable.

Motivated by the desire to design robust learning algorithms, this problem has been studied under various conditions by numerous authors [see, e.g., 3; 11; 8; 7; 9; 10].

We consider two variants of the above model with respect to what observations are available to the learner. In both models the learner can observe its actual state, x_t. In the *full information* feedback model, the learner can observe the full cost function f_t at the end of time instant t (this is equivalent to observing y_t, the uncontrolled part of the state the learner is leaving), while in the *bandit* feedback model the learner only observes the cost $f_t(x_t, a_t)$ it receives.

Treating the online MDP problem as a huge but standard online learning problem, it is relatively not hard to obtain algorithms that enjoy good regret bounds but whose computational complexity is huge. Therefore, earlier work in the literature concentrated on obtaining *computationally efficient* algorithms that also achieve near-optimal regret rates. These results either concern the (stochastic) shortest path problem (SSP, an episodic MDP), or unichain MDPs. Several methods achieve near-optimal regret rates by running an independent expert algorithm for each $x \in \mathcal{X}$, see Even-Dar et al. [2, 3] for the full information case and Neu et al. [8, 7, 9, 10] for the bandit case). Yu et al. [11] gave other low-complexity methods with inferior performance guarantees.

The disadvantage of these methods is that, although they achieve optimal $O(\sqrt{T})$ regret rate in terms of the time horizon T, they often scale suboptimally in other problem parameters, such as the mixing time in the unichain case or the length of the paths in the SSP case. In particular, the optimal-order bounds in the literature for the bandit setting require that all states in $\mathcal{X}$ could be visited with positive probability under *any* deterministic policy, and the inverse of this, potentially very small probability appears in the regret bounds. In this paper we alleviate this problem and obtain optimal-order bounds that do not deteriorate with the minimum visitation probability.

To achieve this, we treat the MDP problem as an online linear optimization problem and show that the resulting methods can be implemented efficiently. We note that the same idea was applied successfully to the deterministic shortest path problem [4], where the minimum visitation probability can also be zero.

2 Preliminaries

First, let us introduce some notation. Let Δ_S denote the set of probability measures over S. Note that for S finite, we can also view as the unit simplex in $\mathbb{R}^{|S|}$: $\Delta_S = \{v \in [0,1]^{|S|} : \sum_{i=1}^{|S|} v_i\}$. The standard inner product of Euclidean spaces will be denoted by $\langle \cdot, \cdot \rangle$. For $p \geq 1$, the p-norm of vector v is denoted by $\|v\|_p$.

The structure of an online MDP is given by a state space $\mathcal{X}$, action spaces $\mathcal{A}(x), x \in \mathcal{X}$, with $\mathcal{U} = \{(x,a) : x \in \mathcal{X}, a \in \mathcal{A}(x)\}$, and probability transition kernel $P : \mathcal{X} \times \mathcal{U} \to [0,1]$ satisfying $\sum_{x \in \mathcal{X}} P(x|u) = 1$ for all $u \in \mathcal{U}$ where $P(x|u) \stackrel{\text{def}}{=} P(u,x)$. The learner's starting state, x_1, is distributed according to some distribution μ_0 over $\mathcal{X}$, where μ_0 is a positive distribution. At each time instant $t = 1, 2, \ldots$, based on its previous observations, state and action sequences, the learner choses an action $a_t \in \mathcal{A}(x_t)$, possibly in a random manner. Extending this notion, we can say that the agent chooses a (randomized) *Markov policy* $\pi_t : \mathcal{U} \to [0,1]$, $\sum_{a \in \mathcal{A}(x)} \pi_t(x,a) = 1$, and choses a_t according to the distribution $\pi_t(x_t, \cdot)$. If $\pi_t = \pi$ independently of t, we say that $\{\pi_t\}$ is stationary and we identify such a control strategy with π. The set of Markov policies will be denoted by Π.

In this paper we will consider two types of MDPs.

Loop-free stochastic shortest path (LF-SSP) problems. Here we assume that $\mathcal{X}$ has a layered structure, that is $\mathcal{X}$ can be partitioned into disjunct sets $\mathcal{X}_1, \ldots, \mathcal{X}_L$ such that if $P(x'|x,a) > 0$ then $x \in \mathcal{X}_l$ and $x' \in \mathcal{X}_{l+1}$ for some $l = 1, \ldots, L-1$, or $x \in \mathcal{X}_L$, $x' \in \mathcal{X}_1$, and $P(x'|x,a) = \mu_0(x')$ for any $a \in \mathcal{A}(x)$. This assumption means that starting in $\mathcal{X}_1$, the learner moves through $\mathcal{X}_2, \mathcal{X}_3, \ldots$ to reach $\mathcal{X}_L$, after which the whole process returns to $\mathcal{X}_1$ and is restarted (we assume without loss of generality that each $x \in \mathcal{X}$ is achievable by following a suitable policy). The transitions from a state in $\mathcal{X}_1$ to a state in $\mathcal{X}_1$ gives rise to an episode of the MDP, and in this case t will index the episodes in the process. Since each episode starts from the same distribution, the episodes are memoryless, and any policy π introduces an occupation measure μ^π over $\mathcal{U}$, where for any stage index l, $\sum_{u \in \mathcal{U}_l} \mu^\pi(u) = 1$, where $\mathcal{U}_I = \{(x,a) : x \in \mathcal{X}_l, a \in \mathcal{A}(x)\}$. Furthermore, for any $x \in \mathcal{X}_1$, $\sum_{a \in \mathcal{A}(x)} \mu^\pi(x,a) = \mu_0(x)$. With this we can view $K = \{\mu^\pi : \pi \in \Pi\}$ as a subset of $\times_{l=0}^{L} \Delta_{\mathcal{U}_l} \subset \times_{l=0}^{L} \mathbb{R}^{|\mathcal{U}_l|} = \mathbb{R}^{|\mathcal{U}|}$. Let $d = |\mathcal{U}|$. Note that K is a convex subset $\mathbb{R}^d$: In fact, is is a polytope as it can be described by a set of linear constraints. Furthermore, with an immediate cost function $f : \mathcal{U} \to [0,1]$, the expected total cost of policy π in an episode can be written as $\langle f, \mu^\pi \rangle$. Note that with this the problem of finding the stationary policy with the smallest per episode expected cost can be written as the linear optimization problem of $\arg\min_{\mu \in K} \langle f, \mu \rangle$: Once the solution of this problem is found, a Markov policy π_μ is extracted from the optimizing measure μ by $\pi_\mu(x,a) = \mu(x,a) / \sum_{a \in \mathcal{A}(x)} \mu(x,a)$. Then, by construction, $\mu^{\pi_\mu} = \mu$. The above description implies that all paths from the starting layer $\mathcal{X}_1$ back to itself are of the same length. This assumption is not restrictive, though, as any layered MDP can be modified without loss of generality to satisfy this assumption [see 4]. For convenience, for online learning with changing costs in LF-SSPs we redefine the regret to be the regret of the first T episodes and use f_t to be the cost function effective in episode t. With this,

$$\mathbf{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} \langle f_t, \mu^{\pi_t} \rangle\right] - \min_{\mu \in K} \sum_{t=1}^{T} \langle f_t, \mu \rangle, \tag{1}$$

where $\pi_t \in \Pi$ is the Markov policy used *in the tth episode*. The problem of keeping the regret low is thus viewed as an instance of *online linear optimization* over the convex set K.

Recurrent MDPs. Here, following previous works, we assume the so-called uniform mixing condition: There exists a number $\tau \geq 0$ such that under any policy π and any pair of distributions μ and μ' over $\mathcal{X}$, $\|(\mu - \mu')P^\pi\|_1 \leq e^{-1/\tau} \|\mu - \mu'\|_1$, where we use the convention of viewing distributions over $\mathcal{X}$ as *row* vectors of $\mathbb{R}^{|\mathcal{X}|}$ and $P^\pi \in \mathbb{R}^{|\mathcal{X}| \times |\mathcal{X}|}$ is the transition probability matrix underlying π: $(P^\pi)_{x,x'} = \sum_{a \in \mathcal{A}(x)} \pi(a|x) P(x'|x,a)$, where WLOG we assume that $\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$. As Even-Dar et al. [3], we call the smallest τ satisfying this assumption the *mixing time* of the transition probability kernel P. This assumption is not unrestrictive, but relaxing it would further complicate the paper and hence we leave this for future work. As for LF-SSPs, for a Markov policy π, let μ^π be its stationary distribution over $\mathcal{U}$. Under

the assumption of $\tau < \infty$, μ^π is uniquely determined. Introduce $K = \{\mu^\pi : \pi \in \Pi\} \subset \Delta_{\mathcal{U}}$. Again, K is a polytope and hence is a convex subset of $\mathbb{R}^{|\mathcal{U}|}$. When discussing recurrent MDPs, we will take $d = |\mathcal{U}|$ as the "dimension" of the problem. In this case, we are concerned with finding a sequence of policies whose expected total cost up to time T is not much larger than that of the best policy in hindsight. Similarly to Neu et al. [7, 10], we can bound this expected additional cost by

$$\mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} f_t(X_t, A_t)\right] - \min_{\pi\in\Pi} \mathbb{E}_\pi\left[\sum_{t=1}^{T} f_t(X_t, A_t)\right] \le \sum_{t=1}^{T} \langle f_t, \mu^{\pi_t} - \mu^\pi\rangle + (\tau+1)Tk + 4\tau + 4, \tag{2}$$

where $\mathbb{E}_{\pi_{t:T}}$ denotes the expectation where the sequence (X_t, A_t) is generated by following the sequence of policies $\pi_1, \dots, \pi_T$ and similarly for $\mathbb{E}_\pi$, and $k \ge \mathbb{E}\left[\|\mu^{\pi_t} - \mu^{\pi_{t+1}}\|_1\right]$for $t = 1, \dots, T$. Since we can recover a sequence of policies from a sequence $\mu_t \in K$, it is enough to find a sequence $\mu_1, \dots, \mu_t \in K$ such that the first term of the bound is small, and k is not too large. This is an online linear optimization problem.

With this, we mapped the problem of online learning in MDPs (in both cases) to online linear optimization, which is well studied problem in online learning (see Bubeck et al. 1 and the references therein).

3 Learning under Full Information in MDPs

In this section, we consider online learning in MDPs when the entire cost vector f_t is observed at each time. The algorithm that we apply is the so-called mirror-descent algorithm with the negative entropy regularizer. To implement the projection to the set K needed by this algorithm, we use gradient descent; we call the resulting composite algorithm the MD^2 algorithm. In order to apply MD^2 we need the components of the (occupation) measures bounded away from zero. This will not be the case, since policies may choose actions with arbitrarily low probabilities. Without loss of generality we can assume that there exists a $\beta > 0$ and a policy π_{exp} such that the corresponding (occupation) measure $\mu_{exp} = \mu^{\pi_{exp}}$ satisfies $\mu_{exp}(x, a) \ge \beta$ for all $(x, a) \in \mathcal{U}$. By the convexity of K, $\mu_\delta = (1-\delta)\mu + \delta\mu_{exp} \in K$ for any $0 < \delta < 1$ and $\mu \in K$ (i.e., there exists a policy inducing μ_δ), and for any loss function f we have $|\langle f, \mu_\delta\rangle - \langle f, \mu\rangle| = \delta|\langle f, \mu_{exp} - \mu\rangle|$. Therefore, we do not loose much if we use MD^2 with $K_{\delta\beta} = \{\mu \in K : \mu(x, a) \ge \delta\beta \text{ for all } x, a\}$ instead of K, since $\mu_\delta \in K_{\delta\beta}$.

First we consider the simple case of the LF-SSP problem. The next result bounds the regret of MD^2 when used with $K_{\delta\beta}$ and the unnormalized negative entropy regularizer $R(\mu) = \sum_{l=0}^{L-1} R_l(\mu_l)$, where $\mu = (\mu_0, \dots, \mu_{L-1})$, $\mu_l \in [0, \infty)^{\mathcal{U}_l}$ and $R_l(\mu_l) = \sum_i \mu_{l,i} \ln(\mu_{l,i}) - \mu_{l,i}$ is the unnormalized negative entropy regularizer over $[0, \infty)^{\mathcal{U}_l}$. We denote by $D(\cdot, \mu') = R(\cdot) - R(\mu') - \langle \nabla R(\mu'), \cdot - \mu'\rangle$ the (unnormalized) Kullback-Leibler divergence underlying R.

Theorem 1. *Assume MD^2 is run for the LF-SSP problem on $K_{\delta\beta}$ with appropriate parameters starting from $\hat{\mu}_1 \in K$. Then, for any $\mu \in K$, the regret of the algorithm can be bounded as $\mathbf{R}_T \le C\sqrt{LD_{\max}T}$ with a universal constant $C > 0$, while the per-step complexity is $O(d^4 T^{1/4} \ln(Td)/\beta)$, where $D_{\max} = \max_{\mu\in K} D_R(\mu, \hat{\mu}_1)$ and $d = |\mathcal{U}|$.*

Note that the regret does not depend on $1/\beta$, but $D_{\max} = \Theta(L \max_l \log(|\mathcal{U}_l|))$, making the regret scale with $O(L\sqrt{T \max_l \log(|\mathcal{U}_l|)})$. Note that Neu et al. [9] gave a $O(L^2\sqrt{T \ln \max_x |\mathcal{A}(x)|})$ for an algorithm whose complexity is $O(d)$ per time-step; the two bounds are incomparable. It is an interesting (and probably challenging) problem to achieve the best of the two results.

In order to apply MD^2 to the recurrent MDP case, we need to obtain a regret bound for the online linear optimization on K and show that the sequence of policies does not change too quickly. We have $\left|\sum_{t=1}^T \langle f_t, \mu\rangle - \sum_{t=1}^T \langle f_t, \mu_\delta\rangle\right| \le \delta \sum_{t=1}^T |\langle f_t, \mu_{exp} - \mu\rangle| \le 2\delta T$. Therefore, running MD^2 on $K_{\delta\beta}$ gives following result:

Theorem 2. *Assume MD^2 is run for the recurrent MDP problem on $K_{\delta\beta}$ with $c = \frac{\beta\delta\eta}{2\sqrt{T}}$, and $\delta = 1/\sqrt{T}$. Then the regret of the sequence of policies given by $\pi_t(x, a) = \frac{\hat{\mu}_t(x,a)}{\hat{\mu}_t(x)}$ relative to any police π is bounded by $\mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^T f_t(X_t, A_t)\right] - \mathbb{E}_\pi\left[\sum_{t=1}^T f_t(X_t, A_t)\right] \le 2\sqrt{TD_R(\mu, \hat{\mu}_1)(1 + 2(\tau+1))} + 3\sqrt{T} + 4\tau + 4$ with a per-step complexity of $O(d^4 T^{1/4} \ln(Td)/\beta)$.*

Note that this improves the previous state-of-the-art bound [7; 10] that scales with $O(\tau^{3/2}\sqrt{T \ln |A|})$ as far as the dependence of the bound on τ is concerned.

4 Learning under Bandit Information in MDPs

The purpose of this section is to consider online learning in MDPs with changing cost functions under bandit feedback, i.e., when at time t, the only information received is $f_t(x_t, a_t)$, the cost of the current transition. Based on the previous section, we see that to control the regret, an MDP learning algorithm has to control the regret in an online linear bandit

problem with decision set K, as well as the rate of change of the policies. As we have already seen, MD-based algorithm will have slowly changing policies, hence our main concern is the linear optimization term.

In this section, focussing on LF-SSPs, we design computationally efficient bandit algorithms based on MD and the continuous exponential weights algorithm. In both cases, the immediate costs will be estimated in the same manner:

$$\widetilde{f}_t(x,a) = f_t(x,a)\mathbb{I}_{\{x_t^{(l)}=x, a_t^{(l)}=a\}}/\mu^{\pi_t}(x,a)\ . \tag{3}$$

Note that in each stage l, $\widetilde{f}_t(x,a)$ is nonzero only for the state-action pair visited in $\mathcal{U}_l$: Hence, $\widetilde{f}_t$ is available to the learner. It is easy to see that as long as *(B) $\mu^{\pi_t}(x,a)$ is bounded away from zero for each state-action pair* (x,a), the above estimate is unbiased. In particular, denoting by $\mathcal{F}_t$ the σ-algebra generated by the history up to the beginning of episode t, $\mathbb{E}\left[\widetilde{f}_t(x,a)|\mathcal{F}_t\right] = f_t(x,a)$ holds for all $(x,a)\in\mathcal{U}$.

As before, we apply MD^2 with the unnormalized negentropy regularizer on $K_{\delta\beta}$. Note that the restriction to $K_{\delta\beta}$ is now used to ensure both that the projection step can be implemented efficiently and that estimates in (3) be well-defined. In particular, this implies that (B) will be satisfied and we get the following result:

Theorem 3. *Run MD^2 on $K_{\delta\beta}$ with the reward estimates* (3)*. Let $\hat{\mu}_t$ be the output of MD^2 in round t, define $\pi_t = \pi_{\hat{\mu}_t}$ (i.e. , by the conditionals induced by $\hat{\mu}_t$) and run π_t in episode t. Then, with appropriate parameters, the regret can be bounded as $\mathbf{R}_T \le C\sqrt{|\mathcal{U}|TD_{\max}}$ with some universal constant $C>0$, while the computational cost is bounded by $O(d^4T^{1/4}\ln(Td)/\beta)$.*

Based on the paper of Narayanan and Rakhlin [6], one can design another algorithm, which we shall call CEWA with Dikin walks, which uses a randomized sampling strategy to sample from the distribution obtained from $q_t(\mu) = q_1(\mu)\exp(-\eta\sum_{s=1}^{t-1}\widetilde{\ell}_s(\mu))$ by normalizing it, where $\widetilde{\ell}_t(\mu) = \langle\widetilde{f}_t,\mu\rangle$ and $\widetilde{f}_t$ is obtained using (3). The following result holds:

Theorem 4. *Let π_t be obtained by CEWA. Then, with appropriate parameters, the regret is bounded by $\mathbf{R}_T \le \sqrt{LD_{\mathrm{KL}}(p_\mu,p_1)T}+1$, while the per step computational complexity is bounded by $O(d^8\ln T)$.*

Note that taking a uniform prior for p_1, $D_{\mathrm{KL}}(p_\mu,p_1) = O(d\ln d)$. Earlier works of Neu et al. [8, 9] assumed that every policy visits every state with positive probability, and their bounds scale inversely with the minimum visitation probability. In contrast, we do not need such an assumption, and our bounds do not scale with any similar constant. The price we pay is a slightly larger computational complexity.

In the recurrent MDP case it is harder to guarantee that an estimate similar to (3) be well-defined. The reason for this is that we need to ensure that each state-action pair can be reached with positive probability when the estimate is formed. To alleviate this problem, one can follow the approach of [10] to form the loss estimates and use the resulting estimates in MD^2.

References

[1] S. Bubeck, N. Cesa-Bianchi, and S.M. Kakade. Towards minimax policies for online linear optimization with bandit feedback. *Journal of Machine Learning Research - Proceedings Track*, 23:41.1–41.14, 2012.

[2] E. Even-Dar, S. M. Kakade, and Y. Mansour. Experts in a Markov decision process. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *NIPS-17*, pages 401–408. MIT Press, 2005.

[3] E. Even-Dar, S.M. Kakade, and Y. Mansour. Online Markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

[4] A. György, T. Linder, G. Lugosi, and Gy. Ottucsák. The on-line shortest path problem under partial monitoring. *Journal of Machine Learning Research*, 8:2369–2403, 2007. ISSN 1532-4435.

[5] J. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors. *NIPS-23*, 2010. Curran Associates.

[6] H. Narayanan and A. Rakhlin. Random walk approach to regret minimization. In Lafferty et al. [5], pages 1777–1785.

[7] G. Neu, A. György, A. Antos, and Cs. Szepesvári. Online Markov decision processes under bandit feedback (extended version). In Lafferty et al. [5].

[8] G. Neu, A. György, and Cs. Szepesvári. The online loop-free stochastic shortest-path problem. In A.T. Kalai and M. Mohri, editors, *COLT 2010*, pages 231–243, 2010.

[9] G. Neu, A. György, and Cs. Szepesvári. The online loop-free stochastic shortest-path problem. *In preparation*, 2013.

[10] G. Neu, A. György, Cs. Szepesvári, and A. Antos. Online Markov decision processes under bandit feedback. *IEEE Transactions on Automatic Control*, 2013. URL `http://www.szit.bme.hu/~gya/publications/NeGySzAn13.pdf`. (accepted for publication).

[11] J.Y. Yu, S. Mannor, and N. Shimkin. Markov decision processes with arbitrary reward processes. *Mathematics of Operations Research*, 34(3):737–757, 2009.

Policy Shaping: Integrating Human Feedback with Reinforcement Learning

Shane Griffith
Department of Computer Science
Georgia Institute of Technology
sgriffith7@gatech.edu

Kaushik Subramanian
Department of Computer Science
Georgia Institute of Technology
ksubrama@cc.gatech.edu

Jonathan Scholz
Department of Computer Science
Georgia Institute of Technology
jkscholz@gatech.edu

Charles L. Isbell
School of Interactive Computing
Georgia Institute of Technology
isbell@cc.gatech.edu

Andrea Thomaz
School of Interactive Computing
Georgia Institute of Technology
athomaz@cc.gatech.edu

Abstract

A long term goal of Interactive Reinforcement Learning is to incorporate non-expert human feedback to solve complex tasks. Some state-of-the-art methods have approached this problem by mapping human information to rewards and values and iterating over them to compute better control policies. In this paper we argue for an alternate and more effective characterization of human feedback: Policy Shaping. We introduce **Advise**, a Bayesian approach that attempts to maximize the information gained from human feedback by utilizing it as direct policy labels.

We compare **Advise** to state-of-the-art approaches using a series of experiments. These experiments use two classic arcade games, together with feedback from a simulated human teacher, which allows us to systematically test performance under a variety of cases of infrequent and inconsistent feedback. We show that **Advise** has similar performance to the state of the art, but is more robust to a noisy signal from the human and fairs well with an inaccurate estimate of its single input parameter. With these advancements this paper may help to make learning from human feedback an increasingly viable option for intelligent systems.

Keywords: Interactive Reinforcement Learning, Human-Agent Interaction, Human-in-the-loop learning

1 Introduction

A long–term goal of machine learning is to create systems that can be interactively trained or guided by non-expert end-users. This paper focuses specifically on integrating human feedback with Reinforcement Learning. One way to address this problem is to treat human feedback as a shaping reward. Yet, recent papers have observed that a more effective use of human feedback is as direct information about policies [1, 2]. Most techniques for learning from human feedback still, however, convert feedback signals into a reward or a value. In this paper we introduce *Policy Shaping*, which formalizes the *meaning* of human feedback as policy feedback, and demonstrates how to use it directly as policy advice. We also introduce **Advise**, an algorithm for estimating a human's Bayes optimal feedback policy and a technique for combining this with the policy formed from Bayesian Q-Learning[1].

We validate our approach using a series of experiments. These experiments use a simulated human teacher and allow us to systematically test performance under a variety of cases of infrequent and inconsistent feedback. The results demonstrate two advantages of **Advise**: 1) it is comparable to or outperforms state of the art techniques for integrating human feedback with Reinforcement Learning; and 2) by formalizing human feedback, we avoid ad hoc parameter settings and make **Advise** robust to infrequent and inconsistent feedback.

2 Reinforcement Learning

Reinforcement Learning (RL) defines a class of algorithms for solving problems modeled as a Markov Decision Process (MDP). An MDP is specified by the tuple (S, A, T, R), which defines the set of possible world states, S, the set of actions available to the agent in each state, A, the transition function $T : S \times A \rightarrow \Pr[S]$, a reward function $R : S \times A \rightarrow \mathbb{R}$, and a discount factor $0 \leq \gamma \leq 1$. The goal of RL is to identify a policy, $\pi : S \rightarrow A$, that maximizes reward.

This paper used an implementation of the Bayesian Q-learning (BQL) RL algorithm [4]. BQL maintains parameters that specify a normal distribution with unknown mean and precision for each Q value, $Q[s, a]$, which represents an estimate of the long-term expected discounted reward for taking action a in state s. This representation approximates the agent's uncertainty in the optimality of each action, which makes the problem of optimizing the exploration/exploitation trade-off straightforward. Because the Normal-Gamma (NG) distribution is the conjugate prior for the normal distribution, the mean and the precision are estimated using a NG distribution with hyperparameters $\langle \mu_0^{s,a}, \lambda^{s,a}, \alpha^{s,a}, \beta^{s,a} \rangle$. These values are updated each time an agent performs an action a in state s, accumulates reward r, and transitions to a new state s'. Details on how these parameters are updated can be found in [4].

The NG distribution for each Q value can be used to estimate the probability that each action $a \in A_s$ in a state s is optimal, which defines a policy, π_R, used for action selection. The optimal action can be estimated by sampling each $\hat{Q}(s, a)$ and taking the max. A large number of samples can be used to approximate the probability an action is optimal by simply counting the number of times an action has the highest Q value [4].

3 Related Work

A key feature of RL is the use of a reward signal. The reward signal can be modified to suit the addition of a new information source (this is known as *reward shaping* [5]). This is the most common way human feedback has been applied to RL. However, several difficulties arise when integrating human feedback signals that may be infrequent, or occasionally inconsistent with the optimal policy–violating the necessary and sufficient condition that a shaping function be potential-based [5]. Another difficulty is the ambiguity of translating a statement like "yes, that's right" or "no, that's wrong" into a reward. Typically, past attempts have been a manual process, yielding *ad hoc* approximations for specific domains. Researchers have also extended reward shaping to account for idiosyncrasies in human input. For example, a drift parameter can account for the human tendency to give less feedback over time [6].

Advancements in recent work sidestep some of these issues by showing human feedback can instead be used as policy feedback. For example, Thomaz and Breazeal [1] added an *UNDO* function to the negative feedback signal, which forced an agent to backtrack to the previous state after its value update. Work by Knox and Stone [2, 7] has shown that a general improvement to learning from human feedback is possible if it is used to directly modify the action selection mechanism of the RL algorithm. Although both approaches use human feedback to modify an agent's exploration policy, they still treat human feedback as either a reward or a value (e.g., "right" becomes $+1$ and "wrong" -1). In our work, we assume human feedback is making a direct statement about the policy itself, rather than influencing the policy through a reward.

4 Policy Shaping

We use feedback labels directly to infer what the human believes is the optimal policy of action in the previous state. We assume a human providing feedback knows the right answer, but noise in the feedback channel introduces inconsistencies between what the human intends to communicate and what the agent observes. Thus, feedback is consistent, $\mathcal{C}$, with the optimal policy with probability $0 < \mathcal{C} < 1$. We also assume that a human watching an agent learn may not provide feedback after every single action, thus the likelihood, $\mathcal{L}$, of receiving feedback has probability $0 < \mathcal{L} < 1$. In the event feedback is received, it is meant as a comment on the optimality of the immediately preceeding action.

Although many different actions may be optimal in a given state, we will assume for this paper that the human knows only one optimal action, which is the one they intend to communicate. In that case, an action, a, is optimal in state s

[1] A longer version of this paper appears in NIPS 2014 [3].

if no other action is optimal. The probability s, a is optimal can be obtained by application of Bayes' rule in conjuction with the binomial distribution and enforcing independence conditions arising from our assumption that there is only one optimal action. This gives: $\mathcal{C}^{\Delta_{s,a}}(1-\mathcal{C})^{\sum_{j\neq a}\Delta_{s,j}}$, where $\Delta_{s,a}$ is the difference between the number of "right" and "wrong" labels. We take this equation to be the probability of performing s, a according to the feedback policy, π_F (i.e., the value of $\pi_F(s,a)$). This is the Bayes optimal feedback policy given the "right" and "wrong" labels seen, the value for $\mathcal{C}$, and that only one action is optimal per state.

Because the use of **Advise** assumes an underlying RL algorithm will also be used, the policies derived from multiple information sources must be reconciled. Before an agent is completely confident in either policy, it has to determine what action to perform using the policy information each provides. We combine the policies from multiple information sources by multiplying them together: $\pi \propto \pi_R \times \pi_F$. Multiplying distributions together is the Bayes optimal method for combining probabilities from (conditionally) independent sources [8]. Note that BQL can only approximately estimate the uncertainty that each action is optimal from MDP reward. Rather than use a different combination method to compensate for the fact that BQL converges too quickly, we introduced the exploration tuning parameter, θ, from [9], that can be manually tuned until BQL performs close to optimal.

5 Experimental Setup

We evaluate our approach using two game domains, Pac-Man and Frogger, with a simulated oracle. Pac-Man consists of a 5x5 grid world with two food pellets, one ghost, walls, and the Pac-Man avatar. The goal is to eat all the pellets while avoiding the ghost. Points are awarded for each pellet (+10) and winning the game (+500). Points are taken away as time passes (-1) and for losing the game (-500). The action set consisted of the four primary cartesian directions. The state representation included Pac-Man's position, the position and orientation of the ghost, and the presence of pellets.

Frogger consists of a 4x4 grid world with two moving cars, two water hazards, and the Frogger avatar. The goal is to cross the road without being run over or jumping into a water hazard. Each car drives one space per time step. The car placement and direction of motion is randomly determined at the start and does not change. As a car disappears off the end of the map it reemerges at the beginning of the road and continues to move in the same direction. The cars moved only in one direction, and they started out in random positions on the road. Each lane was limited to one car. Points are won for arriving at a safe spot on the far side (+500). Points are lost as time passes (-1), for being run over (-500), and for hopping into a water hazard (-500). The action set consisted of the four primary cartesian directions and a stay-in-place action. The state representation included frogger's position and the position of the two cars.

A simulated oracle was used in the place of human feedback, because this allows us to systematically vary the parameters of feedback likelihood, $\mathcal{L}$, and consistency, $\mathcal{C}$ and test different learning settings in which human feedback is less than ideal. The oracle was created manually by a human before the experiments by encoding the optimal action in each state. For states with multiple optimal actions, a small negative reward (-10) was added to the MDP reward of the extra optimal actions to preserve the assumption that only one action be optimal in each state.

6 Experiments

6.1 A Comparison to the State of the Art

In this evaluation we compare Policy Shaping with **Advise** to the more traditional Reward Shaping, as well as recent Interactive RL techniques. Knox and Stone [2, 7] tried eight different strategies for combining feedback with an environmental reward signal and they found that two strategies, *Action Biasing* and *Control Sharing*, consistently produced the best results. Both of these methods convert human feedback to a value but recognize that the information contained in that value is policy information.

Action Biasing, Control Sharing, and Reward Shaping can all be defined using the same set of parameters and variables. Positive and negative feedback is declared a reward r_h, and $-r_h$, respectively. A table of values, $H[s,a]$ stores the feedback signal for s, a. The value $\mathrm{B}[s,a]$ controls the influence of feedback on learning, and is incremented by a constant b when feedback is received for s, a, and is decayed by a constant d at all other time steps.

Action Biasing modifies the action selection of BQL to be $\mathrm{argmax}_a \hat{Q}(s,a) + \mathrm{B}[s,a] * H[s,a]$. Control Sharing defines a transition between π_R and a feedback policy as the probability $P(a = \mathrm{argmax}_a H[s,a]) = min(\mathrm{B}[s,a], 1.0)$[2]. Reward Shaping modifies the MDP reward function to be $R'(s,a) \leftarrow R(s,a) + \mathrm{B}[s,a] * H[s,a]$.

We compared the methods using four different combinations of feedback likelihood, $\mathcal{L}$, and consistency, $\mathcal{C}$, in Pac-Man and Frogger, for a total of eight experiments[3] [4]. Table 1 summarizes the quantitative results. In the ideal case of frequent and correct feedback ($\mathcal{L} = 1.0$; $\mathcal{C} = 1.0$), we see in Table 1 that **Advise** does much better than the other methods early

[2]Control Sharing interprets feedback as a reward, but it does not use that information, so is unaffected if its magnitude changes.

[3]We manually tuned all the parameters before the experiments to maximize MDP reward. BQL:$\langle \mu_0^{s,a} = 0, \lambda^{s,a} = 0.01, \alpha^{s,a} = 1000, \beta^{s,a} = 0.0000\rangle$, $\theta = 0.0001$ for Frogger, and $\theta = 0.5$ for Pac-Man. Discount factor: $\gamma = 0.99$. Action Biasing, Control Sharing, and Reward Shaping: $b = 1$, $d = 0.001$, for Action Biasing $r_h = 100$, and for Reward Shaping $r_h = 100$ in Pac-Man and $r_h = 1$ in Frogger.

[4]We used the conversion $r_h = 1$, 10, 100, or 1000 that maximized MDP reward in the ideal case to also evaluate the three cases of non-ideal feedback. We had to use $r_h = 1.0$ for Reward Shaping in frogger because the agent can end up in infinite loops when feedback is less than ideal. This was not a problem in Pac-Man because the ghost can force Pac-Man out of oscillatory behavior.

	Ideal Case (L = 1.0, C = 1.0)		Reduced Feedback (L = 0.1, C = 1.0)		Reduced Consistency (L = 1.0, C = 0.55)		Moderate Case (L = 0.5, C = 0.8)	
	Pac-Man	Frogger	Pac-Man	Frogger	Pac-Man	Frogger	Pac-Man	Frogger
BQL + Action Biasing	0.24 ± 0.01	0.09 ± 0.03	0.07 ± 0.02	0.02 ± 0.04	-0.14 ± 0.1	0.02 ± 0.04	0.11 ± 0.02	0.05 ± 0.04
BQL + Control Sharing	0.14 ± 0.02	0.04 ± 0.04	0 ± 0.07	0.01 ± 0.04	-1.21 ± 0.07	-0.17 ± 0.08	-0.08 ± 0.11	0.01 ± 0.04
BQL + Reward Shaping	0.23 ± 0.01	0.06 ± 0.04	0.06 ± 0.02	0.02 ± 0.04	-0.2 ± 0.19	0 ± 0.05	0.07 ± 0.08	0.03 ± 0.04
BQL + **Advise**	**0.32 ± 0.01**	**0.23 ± 0.03**	0.09 ± 0.03	**0.09 ± 0.03**	**0 ± 0.06**	0.01 ± 0.04	0.05 ± 0.05	**0.11 ± 0.03**

Table 1: Comparing the learning rates of BQL + **Advise** to BQL + Action Biasing, BQL + Control Sharing, and BQL + Reward Shaping. Each entry represents the average and standard deviation of the cumulative reward in 300 episodes, expressed as the percent of the maximum possible cumulative reward for the domain with respect to the BQL baseline. Negative values indicate performance worse than BQL. Bold values indicate the best performance for that case.

in the learning process. A human reward that does not match both the feedback consistency and the domain may fail to eliminate unnecessary exploration and produce learning rates similar to or worse than RL on its own. **Advise** avoided these issues by not converting feedback into a reward.

The remaining results in Table 1 show performance for each of the non-ideal conditions that we tested: reduced feedback frequency ($\mathcal{L} = 0.1$; $\mathcal{C} = 1.0$), reduced consistency ($\mathcal{L} = 1.0$; $\mathcal{C} = 0.55$), and a case we call moderate ($\mathcal{L} = 0.5$; $\mathcal{C} = 0.8$). Action Biasing and Reward Shaping performed comparably to **Advise** in three. In these three cases the rate of accumulating information from feedback compared to the rate of information gained by BQL was not large enough to make one particular method significantly outperform any other. This is especially true in Pac-Man, because BQL learned to avoid the ghost very quickly, which made finding an optimal policy more a matter of eliminating redundant moves.

The results in Table 1 comprehensively show that **Advise** always performed at or above the BQL baseline, which indicates robustness to less than ideal feedback. In contrast, Action Biasing, Control Sharing, and Reward Shaping blocked learning progress in several cases with reduced consistency (column 5 has the most extreme example). Control Sharing performed worse than BQL in three cases. Action Biasing and Reward Shaping performed worse than BQL in one case.

Having a prior estimate of the feedback consistency, $\mathcal{C}$, allows **Advise** to balance what it learns from the human appropriately with its own learned policy. We could have provided the known value of $\mathcal{C}$ to the other methods, but doing so would not have helped set r_h, b, or d. These parameters had to be tuned since they only slightly correspond to $\mathcal{C}$. We manually selected their values with ideal feedback, and then used those same settings for the other cases. However, different values for r_h, b, and d may produce better results in the cases with reduced $\mathcal{L}$ or $\mathcal{C}$. We tested this next.

6.2 How The Reward Parameter Affects Action Biasing

Here, we test how Action Biasing performed with a range of values for r_h for the case of moderate feedback ($\mathcal{L} = 0.5$ and $\mathcal{C} = 0.8$), and for the case of reduced consistency ($\mathcal{L} = 1.0$ and $\mathcal{C} = 0.55$). Control Sharing was left out of this evaluation because changing r_h did not affect its learning rate. Reward Shaping was left out of this evaluation due to the problems mentioned in Section 6.1. The conversion from feedback into reward was set to either $r_h = 0$, 500, or 1000.

The results in Fig. 1 show that a large value for r_h is appropriate for more consistent feedback; a small value for r_h is best for reduced consistency. This is clear in Pac-Man when a reward of $r_h = 1000$ led to better-than-baseline learning performance in the moderate feedback case, but decreased learning rates dramatically below BQL in the reduced consistency case. In that case, the use of $r_h = 0$ produced the best results. Therefore, r_h depends on feedback consistency.

This experiment also shows that the best value for r_h is somewhat robust to a slightly reduced consistency. A value of either $r = 500$ or 1000, in addition to $r = 100$ (see Table 1), can produce good results with moderate feedback in both Pac-Man and Frogger. The use of a human influence parameter $\mathrm{B}[s, a]$ to modulate the value for r_h is presumably meant to help make Action Biasing more robust to reduced consistency. The value for $\mathrm{B}[s, a]$ is, however, increased by b whenever feedback is received, and reduced by d over time; b and d are more a function of the domain than the information in accumulated feedback. Our next experiment demonstrates why this is bad for IRL.

6.3 How Domain Size Affects Learning

Action Biasing, Control Sharing, and Reward Shaping use a 'human influence' parameter, $\mathrm{B}[s, a]$, that is a function of the domain size more than the amount of information in accumulated feedback. To show this we froze the parameters and evaluated the algorithms in a larger domain. Frogger was increased to a 6×6 grid with four cars. An oracle was created automatically by running BQL to 50,000 episodes 500 times, and then for each state choosing the action with the highest value. The oracle provided moderate feedback ($\mathcal{L} = 0.5$; $\mathcal{C} = 0.8$) for the 33360 different states identified in this process.

Our results (omitted due to space constraints) show that, whereas **Advise** performed roughly the same as in the smaller Frogger domain (see the last column in Table. 1), Action Biasing, Control Sharing, and Reward Shaping all had a negligible effect on learning, performing roughly the same as the BQL baseline. In order for those methods to perform as well as they did with the smaller version of Frogger, the value for $\mathrm{B}[s, a]$ needs to be set higher and decayed more slowly by manually finding new values for b and d. Thus, like r_h, the optimal values to b and d are dependent on both the domain and the quality of feedback. In contrast, the estimated feedback consistency, $\hat{\mathcal{C}}$, used by **Advise** only depends on the true feedback consistency, $\mathcal{C}$. For comparison, we next show how sensitive **Advise** is to a suboptimal estimate of $\mathcal{C}$.

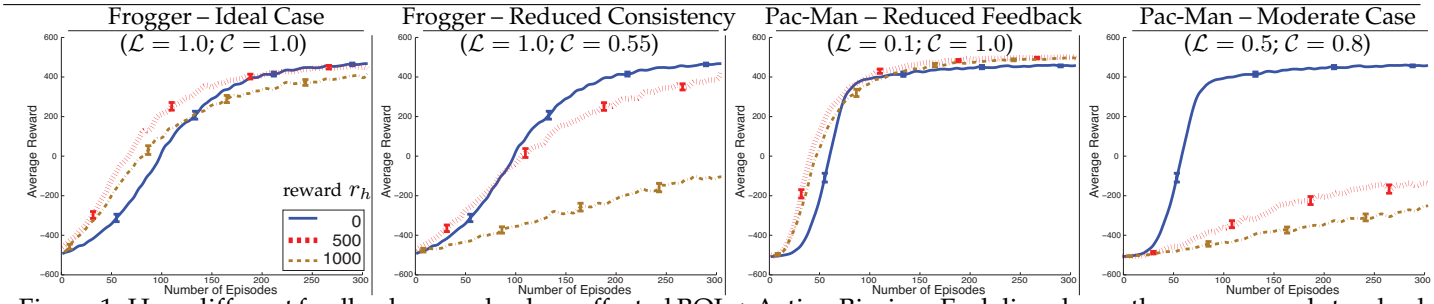


Figure 1: How different feedback reward values affected BQL + Action Biasing. Each line shows the average and standard error of 500 learning curves over a duration of 300 episodes. See the text for more details.

6.4 Using an Inaccurate Estimate of Feedback Consistency

Interactions with a real human will mean that in most cases **Advise** will not have an exact estimate, $\hat{\mathcal{C}}$, of the true feedback consistency, $\mathcal{C}$. It is presumably possible to identify a value for $\hat{\mathcal{C}}$ that is close to the true value. Any deviation from the true value, however, may be detrimental to learning. This experiment shows how an inaccurate estimate of $\mathcal{C}$ affected the learning rate of **Advise**. Feedback was generated with likelihood $\mathcal{L} = 0.5$ and a true consistency of $\mathcal{C} = 0.8$. The estimated consistency was either $\hat{\mathcal{C}} = 1.0$, 0.8, or 0.55.

Our results (omitted due to space constraints) show that in both Pac-Man and Frogger using $\hat{\mathcal{C}} = 0.55$ reduced the effectiveness of **Advise**. The learning curves are similar to the baseline learning curves because using an estimate of $\mathcal{C}$ near 0.5 is equivalent to not using feedback at all. In general, values for $\hat{\mathcal{C}}$ below $\mathcal{C}$ decreased the possible gains from feedback. In contrast, using an overestimate of $\mathcal{C}$ slightly boosted learning rates for these particular domains and case of feedback quality. In general, however, overestimating $\mathcal{C}$ can lead to a suboptimal policy especially if feedback is provided very infrequently. Therefore, it is desirable to use $\hat{\mathcal{C}}$ as close to its true value, $\mathcal{C}$, as possible.

7 Conclusion and Future Work

Overall, our experiments indicate that it is useful to interpret feedback is as a direct comment on the optimality of an action, without converting it into a reward or a value. **Advise** performed comparably to or better than tuned versions of Action Biasing, Control Sharing, and Reward Shaping. These methods are outperformed because they first convert feedback into a reward, which reduces the effectiveness of the information. Their performance also suffers because their use of 'human influence' parameters is disconnected from the amount of information in the accumulated feedback. In contrast, **Advise** has only one input parameter, which is independent of the domain, and can be used to calculate the exact amount of information in the accumulated feedback in each state. **Advise** combines the feedback policy with the RL policy using the right amount of influence. It also always utilizes information from both sources.

In conclusion, this paper defined the Policy Shaping paradigm for integrating feedback with Reinforcement Learning. We introduced **Advise**, which tries to maximize the utility of feedback using a bayesian approach to learning. **Advise** produced results on par with or better than the current state-of-the-art IRL techniques, showed where those approaches fail while **Advise** is unaffected, and it demonstrated robustness to infrequent and inconsistent feedback. We plan to extend our work by computing $\hat{\mathcal{C}}$ online as a human interacts with an agent, and addressing other aspects of human feedback like errors in credit assignment.

References

[1] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, 2008.

[2] W. B. Knox and P. Stone, "Combining manual feedback with subsequent MDP reward signals for reinforcement learning," in *AAMAS*, pp. 5–12, 2010.

[3] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. Thomaz, "Policy Shaping: Integrating Human Feedback with Reinforcement Learning," in *NIPS*, 2014.

[4] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *AAAI*, pp. 761–768, 1998.

[5] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, pp. 341–348, 1999.

[6] C. L. Isbell, C. Shelton, M. Kearns, S. Singh, and P. Stone, "A social reinforcement learning agent," in *Agents*, pp. 377–384, 2001.

[7] W. B. Knox and P. Stone, "Reinforcement learning from simultaneous human and MDP reward," in *AAMAS*, pp. 475–482, 2012.

[8] C. Bailer-Jones and K. Smith, "Combining probabilities." GAIA-C8-TN-MPIA-CBJ-053, July 2011.

[9] T. Matthews, S. D. Ramchurn, and G. Chalkiadakis, "Competing with humans at fantasy football: Team formation in large partially-observable domains," in *AAAI*, pp. 1394–1400, 2012.

Dopamine D2 Receptor Availability Associated with Probabilistic Reward Learning

Jacob S. Young
Pritzker School of Medicine
University of Chicago
youngjs@uchicago.edu

Gregory R. Samanez-Larkin
Department of Psychology
Yale University
g.samanezlarkin@yale.edu

David H. Zald
Department of Psychology
Vanderbilt University
david.zald@vanderbilt.edu

Abstract

A wealth of prior research has implicated the neurotransmitter dopamine in reinforcement learning and decision making. However, few studies have examined how individual differences in human reinforcement learning may be related to individual differences in the dopamine system, and no studies have previously used PET imaging of the dopamine system to examine individual differences in reinforcement learning in humans. A sample of 25 healthy young adults completed a reinforcement learning task and a [^{18}F]Fallypride PET scan of dopamine D2/D3 receptors. A whole-brain analysis revealed an association between striatal D2 receptors and reinforcement learning such that individuals with higher levels of receptor availability in the right ventromedial caudate and nucleus accumbens were better able to learn from probabilistic feedback which of two stimuli had a higher expected value. The task included both gain and loss conditions, but the effects were not specific to either condition and instead were related to general learning ability. To our knowledge this is the first study to demonstrate an association between a direct measure of the human dopamine system and reinforcement learning. Consistent with a large body of prior animal work, our results suggest that the human striatal dopamine system promotes reinforcement learning.

Keywords: dopamine, reinforcement learning, individual differences, ventromedial caudate, nucleus accumbens

Acknowledgements

This research was supported by National Institute of Drug Abuse grant DA033611 to D.H.Z. G.R.S-L. was supported by Pathway to Independence Award AG042596.

Extended Abstract

Learning from rewards and aversive stimuli is necessary for everyday life. Consistently, studies have found that the nucleus accumbens (NAcc), which is innervated by midbrain dopamine neurons, is recruited during the anticipation of reward and encodes reward prediction errors [1-4]. Individual differences in the prediction error signal in the striatum during reinforcement-based learning tasks distinguish learners from non- learners [5]. Furthermore, enhancing the dopamine system pharmacologically with L-DOPA has been shown to improve choice performance towards monetary gains but not monetary losses [6]. In a related study, treating older adults with L-DOPA improved reward prediction errors in the NAcc especially for individuals who had abnormal signaling in the NAcc at baseline [7]. These results suggest a crucial role for individual differences in dopamine transmission, particular in the NAcc, contributing to behavioral differences in reinforcement learning in response to reward feedback.

Twenty-five healthy, young adult participants (61% female; age range: 18–24, M=20.88) completed our study at Vanderbilt University. All participants were free of current or past medical or psychiatric illness, drug free, and in good health.

Participants were instructed to repeatedly choose between a pair of abstract images (Fig. 1) in a stationary two-armed bandit task. On each trial, participants chose from one of three pairs of fractal cues corresponding to three conditions: gain, loss, neutral [8]. Images in the gain cue pair each had a chance of winning $1, but one image would win 66% of the time while the other image would win 33% of the time. The loss cue pair each had a chance of losing $1, but one image would lose 66% of the time and the other image would lose 33% of the time. In the neutral cue pair all choices yielded a $0 outcome. Participants were instructed to try to win as much money as possible (i.e., maximize their earnings) and that they would be paid based on their performance. Participants completed a practice run which included 24 trials of each condition (i.e. gain, loss, neutral) for a total of 72 trials. Participants then completed two paid runs, each with the same parameters as the practice run. The participants received feedback on their current trial earnings (but not cumulative earnings) after each choice. On a separate visit, the participants received a 5 mCi injection of [18F]Fallypride and underwent a PET scan to assess D2-like dopamine receptor availability.

There was no difference between the percent of correct choices on gain trials (77.77%; range: .45–1.00) and the percent of correct choices on loss trials (78.42%; range: .56-.97) (t_{24} = –.2; p = .84). The average percent of choices of the higher expected value cue in the learning task (across conditions) was associated with greater [18F]Fallypride binding in the right ventromedial caudate and nucleus accumbens (peak t_{24} = 4.62; max coordinates x = –4; y = 10; z = –6) after controlling for age and gender (Fig. 2). We extracted binding potential values from this cluster to compare valence conditions. The correlation between D2 binding and gain learning was not significantly different than the correlation between D2 binding and loss learning, t = –0.27, p = .60. These data run counter to a prediction that there would be a stronger correlation with gain learning based on the pharmacological findings mentioned above. However it is important to note that we cannot distinguish learning from positive and negative *PEs* in this task; both positive and negative PEs are generated in the gain and loss conditions.

Previously it has been demonstrated in monkeys that individual differences in D2 receptor availability are associated specifically with the ability to shift behavior in a reversal learning task [9]; this effect was limited to the dorsal striatum and valence-specific. We also collected behavioral data from a reversal learning task, but did not observe these effects in our human sample.

To our knowledge, this is the first study to use human PET imaging to directly measure individual differences in dopamine receptors and reinforcement learning. The findings suggest that the level of available D2/D3 receptors in the NAcc and ventromedial caudate is associated with improved learning from feedback in general. Higher levels of D2 receptor availability may provide more sites for dopamine to act promoting enhanced reinforcement learning. Consistent with prior animal work, our results suggest that the human striatal dopamine system promotes reinforcement learning. Individual differences in the dopamine system may be associated with learning abilities that influence broader decision making in everyday life, and our results suggest that dopaminergic drugs may enhance decision making where learning rapidly from novel feedback is required.

References

1. Knutson B, Adams CM, Fong GW, Hommer D: **Anticipation of increasing monetary reward selectively recruits nucleus accumbens**. *J Neurosci* 2001, **21**(16):RC159.
2. Clithero JA, Reeck C, Carter RM, Smith DV, Huettel SA: **Nucleus accumbens mediates relative motivation for rewards in the absence of choice**. *Front Hum Neurosci* 2011, **5**:87.
3. McClure SM, Berns GS, Montague PR: **Temporal prediction errors in a passive learning task activate human striatum**. *Neuron* 2003, **38**(2):339-346.
4. Haber SN, Fudge JL, McFarland NR: **Striatonigrostriatal pathways in primates form an ascending spiral from the shell to the dorsolateral striatum**. *J Neurosci* 2000, **20**(6):2369-2382.
5. Schönberg T, Daw ND, Joel D, O'Doherty JP: **Reinforcement learning signals in the human striatum distinguish learners from nonlearners during reward-based decision making**. *The Journal of Neuroscience* 2007, **27**(47):12860-12867.
6. Pessiglione M, Seymour B, Flandin G, Dolan RJ, Frith CD: **Dopamine-dependent prediction errors underpin reward-seeking behaviour in humans**. *Nature* 2006, **442**(7106):1042-1045.
7. Chowdhury R, Guitart-Masip M, Lambert C, Dayan P, Huys Q, Düzel E, Dolan RJ: **Dopamine restores reward prediction errors in old age**. *Nature neuroscience* 2013.
8. Knutson B, Samanez-Larkin GR, Kuhnen CM: **Gain and loss learning differentially contribute to life financial outcomes**. *PloS one* 2011, **6**(9):e24390.
9. Groman SM, Lee B, London ED, Mandelkern MA, James AS, Feiler K, Rivera R, Dahlbom M, Sossi V, Vandervoort E *et al*: **Dorsal striatal D2-like receptor availability covaries with sensitivity to positive reinforcement during discrimination learning**. *J Neurosci* 2011, **31**(20):7291-7299.

Figures

Figure 1. *Task Description*. Note that a neutral condition is not shown.

Figure 2. Correlation between dopamine receptor availability and reinforcement learning.