Learning and Planning with Timing Information in Markov Decision Processes

Pierre-Luc Bacon, Borja Balle and Doina Precup
Reasoning and Learning Lab
McGill University
Montreal, Canada
{pbacon,bballe,dprecup}@cs.mcgill.ca

Abstract

We consider the problem of learning and planning in Markov decision processes with temporally extended actions represented in the options framework. We propose to use predictions about the duration of extended actions to represent the state and show that this leads to a compact predictive state representation model independent of the set of primitive actions. Then we develop a consistent and efficient spectral learning algorithm for such models. Using just the timing information to represent states allows for faster improvement in the planning performance. We illustrate our approach with experiments in both synthetic and robot navigation domains.

Keywords: Predictive State Representations, Options, Planning

Acknowledgements

This work was supported by the Fonds Quebecois de la Recherche sur la Nature et les Technologies (FQRNT) and the Natural Sciences and Engineering Research Council (NSERC)

1 Introduction

Modelling the dynamics of an agent embedded in a large, complex environment is key to building good planning algorithms for such agents. In most practical applications, models are carefully designed by hand, and the agent's "state" is given by measurements which are understandable by the designer of the system (such as spatial location and velocity, in the case of a robot). However, learning dynamical models for such states from data, as well as planning with them can be quite tricky. An alternative idea is to use models that are "subjective", centered on the agent's own perception and action capabilities. For example, affordances [Gibson, 1977] describe "state" through the courses of action that are enabled. Similarly, in robotics, subjective representations have been used to model dynamics, e.g. [Bowling *et al.*, 2005; Stober *et al.*, 2011]. Such models are appealing from a psychological point of view, but run into computational problems in very large observation spaces.

In this paper, we focus on a special class of subjective models, *timing models*, which arise from restricting the observations available to the agent to just information about the duration of certain courses of action. Timing of events is understood to be crucial to animal learning [Machado *et al.*, 2009]. The goal of this paper, however, is not learning of the timing of external events, but rather to learn the duration of courses of action that an agent might take. The ensemble of such durations will constitute the agent's *state*, which will be maintained as new data is received. We use the framework of options [Sutton *et al.*, 1999] to model extended courses of actions, and we present an approach for learning option durations.

Our models over durations can be viewed as affordances if we consider an option to be available if its estimated duration is within some reasonable bounds. Note that these models are much simpler than full option models, which provide joint information on the timing as well as the state or observation in which the option will terminate, e.g. [Wolfe and Singh, 2006]. Our approach can also be interpreted as a computationally and statistically efficient way of exploiting prior information about useful courses of action provided in the form of options. As a consequence, the size of our models is independent of the number of possible primitive actions in the underlying system. Another interesting feature of our approach is that we are able to learn feature representations for states using timing information only; this means our method can be applied to observable settings with high-dimensional observations and to partially observable settings as well.

Of course, the utility of such timing models depends strongly on the nature of the task to be solved by the agent, as well as on the "quality" of the options available to the agent. The simplest example in which option duration models are beneficial is that of minimum time to goal problems, in which an agent receives a fixed penalty per time step until its task is completed. In this case, knowing the duration of an option immediately gives us the reward model, so the option duration model has direct value for a planner. More generally, option duration models are beneficial as a form of localization. If you imagine a robot that has models for options that move radially out from the current position, this would allow localizing with respect to all neighboring walls. Finally, consider a problem in which a financial agent is holding stocks, and options which hold a particular stock while it is above a certain value, and sell under that value. In this case, timing models tell us exactly when stocks would be crossing certain barriers. It is clear in this case that, even though we are estimating only durations, these encode important state information (because of the way in which the options are defined).

In this paper we analyze the capacity of option duration models to represent states in a Markov Decision Process (MDP). We propose a spectral algorithm for learning option duration models which builds on existing work for learning transformed predictive state representations [Rosencrantz *et al.*, 2004a]. Finally we evaluate the quality of learning and planning with our model in experiments with discrete MDPs.

1.1 Markov Decision Processes and Temporally Extended Actions

A *Markov decision process* (MDP) is a tuple $M = \langle S, A, P, R \rangle$ where S is the state space, A is the action set, $P : S \times A \to (S \to [0, 1])$ defines a probability distribution over next states, and $R : S \times A \to \mathbb{R}$ is the expected reward function (see [Puterman, 1994] for a review). We refer to probability distributions on S by α, but sometimes use $\boldsymbol{\alpha}$ to stress that we view them as vectors in $\mathbb{R}^S$. Suppose α is a distribution over S and $\pi : S \times A \to [0, 1]$ is a stochastic action policy which, given state s, chooses action a with probability $\pi(s, a)$. The environment then returns a state sampled from P; and the resulting state distribution α' is given by:

$$\alpha'(s') = \sum_{s \in S} \alpha(s) \sum_{a \in A} \pi(s, a) P(s, a)(s') \ . \tag{1}$$

Temporal abstraction in MDPs has been used as a tool to speed up learning and planning algorithms. We adopt the framework of options [Sutton *et al.*, 1999], with the goal of learning state representations based on option timing models. An *option* is a tuple $\omega = \langle I_\omega, \pi_\omega, \beta_\omega \rangle$ where $I_\omega \subseteq S$ is the set of initial states, $\pi_\omega : S \times A \to [0, 1]$ is the option's stochastic action policy, and $\beta_\omega : S \to [0, 1]$ is the option termination probability for each state.

1.2 Predictive State Representations

A *predictive state representation* is a model of a dynamical system where the current state is represented as a set of predictions about the future behavior of the system [Littman *et al.*, 2002; Singh *et al.*, 2004]. We use a particular instantiation of this general idea, the so-called *transformed linear predictive state representation* [Rosencrantz *et al.*, 2004b], which we abbreviate for simplicity as PSR.

A PSR with observations in a finite set Σ is a tuple $\mathcal{A} = \langle \boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}_{\sigma\in\Sigma}\rangle$ where $\boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$ are the initial and final weights respectively, and $\mathbf{A}_\sigma \in \mathbb{R}^{n\times n}$ are the transition weights. The dimension n of these vectors and matrices is the *number of states* of the PSR. The function $f_\mathcal{A} : \Sigma^\star \to \mathbb{R}$ computed by $\mathcal{A}$ assigns a number to each string $x = x_1 x_2 \cdots x_t \in \Sigma^\star$ as follows:

$$f_\mathcal{A}(x) = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_{x_1}\mathbf{A}_{x_2}\cdots\mathbf{A}_{x_t}\boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty \ . \tag{2}$$

The behavior of a stochastic dynamical system producing observations in a finite set Σ can be entirely characterized by the function $f : \Sigma^\star \to \mathbb{R}$ giving the probability $f(x)$ of observing each possible sequence of observations x. A convenient algebraic way to summarize all the information conveyed by f is its *Hankel matrix*, a bi-infinite matrix $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star\times\Sigma^\star}$ with rows and columns indexed by strings in $\Sigma^\star$. In particular, a well-known result states that $\mathbf{H}_f$ has rank at most n if and only if there exists a PSR $\mathcal{A}$ with n states satisfying $f_\mathcal{A} = f$ [Carlyle and Paz, 1971; Fliess, 1974]. The Hankel matrix $\mathbf{H}_f$ is tightly related to the *system dynamics matrix* (SDM) of the stochastic process described by f [Singh *et al.*, 2004], but while the entries of the Hankel matrix represent *joint* probabilities over prefixes and suffixes, the corresponding entry in the SDM is the *conditional* probability of observing a suffix given the prefix. An empirical estimate of the Hankel matrix can be obtained given a finite set of prefixes and suffixes. The singular value decomposition can then be used to recover a PSR (see [Boots *et al.*, 2011] for details).

2 Option Duration Models

We are interested in the dynamics of an agent interacting with an MDP M via a set of options Ω. Recall that in this setting the agent is not allowed to perform primitive actions, and options must be executed until termination. We are interested in considering situations where the *duration* of an option constitutes an informative statistic about the state of the MDP. Hence, the history of the agent's interaction with an MPD will be given by a trajectory consisting of option-duration pairs: $(\omega_1, d_1)(\omega_2, d_2)\cdots(\omega_t, d_t)$, with $\omega_i \in \Omega$, $d_i \in \mathbb{N} = \{1, 2, \ldots\}$. Focusing on the sequence of options and termination/continuation events, we have a discrete dynamical system with observations from $\Omega \times \{\sharp, \bot\}$, where $\sharp$ (*sharp*) denotes continuation and $\bot$ (*bottom*) denotes termination. The previous trajectory in this new dynamical system looks as follows:

$$(\omega_1, \sharp, \ldots, \omega_1, \sharp, \omega_1, \bot, \omega_2, \sharp, \ldots, \omega_2, \sharp, \omega_2, \bot, \ldots) = (\omega_1, \sharp)^{d_1-1}(\omega_1, \bot)(\omega_2, \sharp)^{d_2-1}(\omega_2, \bot)\ldots$$

Formally, we are mapping a dynamical process with trajectories in $(S \times A)^\star$ (representing the interaction of the agent with the MDP), to a process with trajectories in $(\Omega\times\{\sharp, \bot\})^\star$ representing the duration of option execution. This mapping induces a new dynamical system, whose properties might be useful for planning with options in the original system.

We now show that the probability distributions over the duration of options can be compactly represented in the form of a PSR. Let $s_0 \in S$, $\omega = \langle I, \pi, \beta\rangle$, and $d > 0$ be an integer. We write $\delta(s_0, \omega)$ for the random variable representing the duration until termination of option ω from state s_0. We are interested in the following quantity:

$$\mathbb{P}[\delta(s_0, \omega) = d] = \sum_{\bar{s}\in S^d}\sum_{\bar{a}\in A^d} \mathbb{P}[s_0, a_0, s_1, a_1, \cdots, a_{d-1}, a_{d-1}, s_d, \bot] \ , \tag{3}$$

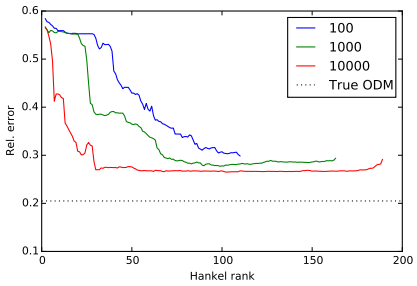
where $\bar{s} = s_1 \cdots s_d$ is the sequence of states traversed by ω, $\bar{a} = a_0 \cdots a_{d-1}$ is the sequence of actions performed by by ω, and $\bot$ denotes the option termination. With some algebra, it can be shown that summing this expression over $\bar{s}$ and $\bar{a}$ yields:

$$\mathbb{P}[\delta(s_0, \omega) = d] = \mathbf{e}_{s_0}^\top \mathbf{A}_{\omega,\sharp}^{d-1}\mathbf{A}_{\omega,\bot}\mathbf{1} \ , \tag{4}$$

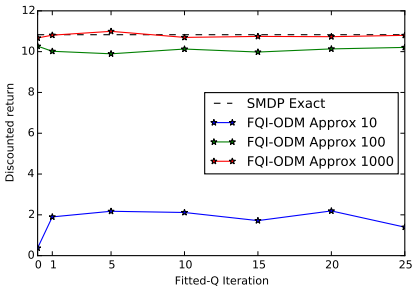
where we have used the following definitions: $\mathbf{e}_{s_0} \in \mathbb{R}^S$ is an indicator vector with $\mathbf{e}_{s_0}(s) = \mathbb{I}[s = s_0]$, $\mathbf{A}_{\omega,\sharp} \in \mathbb{R}^{S\times S}$ is a matrix with $\mathbf{A}_{\omega,\sharp}(s, s') = \sum_{a\in A}\pi(s, a)P(s, a, s')(1 - \beta(s'))$, $\mathbf{A}_{\omega,\bot} \in \mathbb{R}^{S\times S}$ is a matrix with $\mathbf{A}_{\omega,\bot}(s, s') = \sum_{a\in A}\pi(s, a)P(s, a, s')\beta(s')$, and $\mathbf{1} \in \mathbb{R}^S$ is a vector of ones. More generally, we can prove the following:

Theorem 1. *Let M be an MDP with n states, Ω a set of options, and $\Sigma = \Omega \times \{\sharp, \bot\}$. For every distribution α over the states of M, there exists a PSR $\mathcal{A} = \langle \boldsymbol{\alpha}, \mathbf{1}, \{\mathbf{A}_\sigma\}\rangle$ with at most n states that computes the distributions over durations of options executed from a state sampled according to α.*

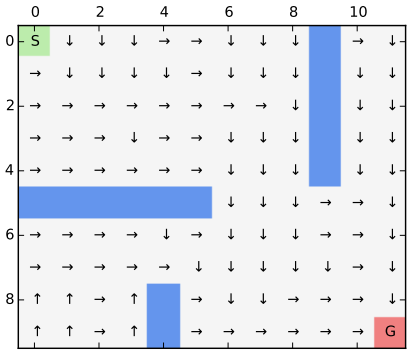
We will call any PSR computing distributions over durations of options an *option duration model* (ODM).



(a) Predictive accuracy: relative error versus rank for different sample sizes



(b) Planning with a learned ODM



(c) Room environment overlaid with an optimal policy over options

3 Experiments

We first assess the learnability of our model in practice using a gridworld environment. We use a 4-connected grid with four actions representing the cardinal directions (NEWS). Unless the current state is a "wall" each action moves the agent one step in the specified direction with probability 0.9, and remains in the current state with probability 0.1. We also define one option for each cardinal direction. These options take as many steps as possible in the specified direction until they hit a wall, at which point the option terminates. A uniform random exploration policy is used for sampling 10000 episodes in which five options are executed up to termination. We also collected a test set consisting of 10000 trajectories of eight options sequences. We evaluate the prediction accuracy by computing the relative error over the estimated remaining number of steps in the currently executing option. For each test trajectory, we picked a time index uniformly at random and conditioned the learned ODM on the history up to this point. These random split points were then kept fixed throughout all evaluations. Figure 1a shows that the prediction accuracy increases as the dimension of the ODM gets larger. More samples also allow for better predictions. Note that since the prediction task is inherently stochastic, even the true ODM cannot achieve zero relative error.

3.1 Planning

We use the Fitted-Q iteration (FQI) algorithm of Ernst *et al.* [2005] for planning over a learned ODM. We make state directly over the ODM state vector updated at each step with the corresponding operator (continuation or termination) according to the linear form in (2). A gridworld environment with obstacles is used for evaluation and once again, any of the four actions can fail with probability 0.1 in every state. An immediate reward of 100 is obtained at the goal and collisions with the walls are penalized by -10. Taking a primitive step does not incur an immediate cost but the length of the trajectories affect the cumulative reward through a discount factor of 0.9. A dataset of 1000 trajectories of eight options sequences was collected with a discrete uniform policy over options. For each curve shown in 1b, we use our dataset to learn an ODM and plan over it. We evaluate the performance of the greedy policy by taking 100 Monte-Carlo estimates in the simulated environment. Given the true underlying MDP and a set of options, we can compute the resulting Semi-Markov Decision Process (SMDP) (see p. 26 of Sutton *et al.* [1999]) and solve it using value iteration. The expected discounted cumulative return in the SMDP serves as our baseline. Figure 1b shows that an optimal policy can be obtained using 1000 trajectories and one step of FQI. Interestingly, it seems that even when using an imperfect model (such as the one built with 100 trajectories in fig. 1b), we can still recover a near-optimal policy.

References

B. Boots, S. Siddiqi, and G. Gordon. Closing the learning planning loop with predictive state representations. *International Journal of Robotic Research*, 2011.

M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. *International Conference on Machine Learning (ICML)*, 2005.

J. W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computer Systems Science*, 1971.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.*, 6:503–556, December 2005.

M. Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 1974.

J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing*, 1977.

M.L. Littman, R.S. Sutton, and S. Singh. Predictive representations of state. *Neural Information Processing Systems (NIPS)*, 2002.

A. Machado, M. T. Malheiro, and W. Erlhagen. Learning to time: A perspective. *Journal of the Experimental Analysis of Behavior*, 2009.

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

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. *International Conference on Machine Learning (ICML)*, 2004.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. *International Conference on Machine Learning (ICML)*, 2004.

S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. *Uncertainty in Artificial Intelligence (UAI)*, 2004.

J. Stober, R. Miikkulainen, and B. Kuipers. Learning geometry from sensorimotor experience. *Joint Conference on Development and Learning and Epigenetic Robotics*, 2011.

R. S Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

B. Wolfe and S. Singh. Predictive state representations with options. *International Conference on Machine Learning (ICML)*, 2006.