Predictive Timing Models

Pierre-Luc Bacon
McGill University
pbacon@cs.mcgill.ca

Borja Balle
McGill University
bballe@cs.mcgill.ca

Doina Precup
McGill University
dprecup@cs.mcgill.ca

Abstract

We consider the problems 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 how 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.

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 [6] describe "state" through the courses of action that are enabled. Similarly, in robotics, subjective representations have been used to model dynamics, e.g. [2, 13]. 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 predictive models, *timing models*. Timing of events is understood to be crucial to animal learning [9]. 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 [14] to model extended courses of actions, and we present an approach for learning option durations. Note that such models can be viewed as affordances, if we consider an option to be available if its estimated duration is within some reasonable bounds. At the same time, 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. [15].

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 neighbouring 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 propose a spectral learning algorithm for option duration models, which builds on existing work for learning transformed predictive state representations [10], and show that it learns useful predictions with reasonable amounts of data. Proofs of mathematical statements are omitted due to space limitations and will appear in a longer version of this work.

2 Modelling Option Duration with PSR

We are interested in the dynamics of an agent interacting with an MDP via a set of options Ω, from the point of view of the options' termination events. Hence, trajectories consist of a sequence of options and their durations $(\omega_1, d_1, \omega_2, d_2, \ldots)$. Focusing on the sequence of options and termination/continuation events, we have a discrete dynamical system with observations from $\Omega \times \{\sharp, \perp\}$, where $\sharp$ (*sharp*) denotes continuation and $\perp$ (*bottom*) denotes termination. The previous trajectory in this new dynamical system looks as follows:

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

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 $(\Omega \times \{\sharp, \perp\})^\star$ representing the duration of option execution.

Given $s \in S$ and an option $\omega \in \Omega$, let $\delta(s, \omega)$ be the random variable counting the number of steps until termination when following ω from s. Note that s might be an initial state for ω, but also a state traversed during its execution, in which case $\delta(s, \omega)$ is the remaining duration until termination.

Let $\mathcal{D}$ denote the set of all probability distributions over $\mathbb{N}$, and $D_\omega^s \in \mathcal{D}$ the distribution of the random variable $\delta(s, \omega)$. Thinking of this distribution as a function of s, we define a map $\Delta_\omega : S \to \mathcal{D}$, with $\Delta_\omega(s) = D_\omega^s$. Bundling together the maps $\{\Delta_\omega\}_{\omega\in\Omega}$, we get a map $\Delta : S \to \mathcal{D}^\Omega$ that assigns to each state in the MDP the tuple of distributions over the durations of all options available to the agent[1]. These maps extend directly to probability distributions over states: if α is a distribution on S, then $\Delta_\omega(\alpha) = \sum_{s\in S} \alpha(s)\Delta_\omega(s)$, which is a mixture of probability distributions from $\mathcal{D}$. The extension of Δ to distributions is immediate.

We say that the set of options Ω is *rich* for MDP M if the map $\Delta : S \to \mathcal{D}^\Omega$ is injective, so for every $s, s' \in S$ there exists some option $\omega \in \Omega$ such that $D_\omega^s \neq D_\omega^{s'}$. Clearly, in this case, Δ will uniquely identify states. If Δ is not injective, such models can still be sufficient for planning in special circumstances. For example, consider minimum-time-to-goal problems, in which fixed negative rewards are attributed per time step, and suppose the agent intends to plan using options only. In this case, states for which $\Delta(s) = \Delta(s')$ will also have the same optimal value function V_Ω^* (a result that follows directly form the way in which option models are defined [14]).

We will now establish a predictive state representation for option duration models. 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 [8, 12]. We use a particular instantiation of this general idea, the so-called *transformed linear predictive state representation* [11], 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. Though PSR is the usual name for this type of model in the reinforcement learning literature, they are also called *weighted finite automaton* (WFA) [5] or *observable operator model* (OOM) [7]. This distinction reflects the fact that this same parametrization can be used to define models with different semantics, depending on the meaning associated to the values computed by the PSR.

A PSR $\mathcal{A}$ computes a function $f_\mathcal{A} : \Sigma^\star \to \mathbb{R}$ that assigns a number to each string $x = x_1x_2\cdots x_t \in \Sigma^\star$ as follows:

$$f_\mathcal{A}(x) = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_{x_1}\mathbf{A}_{x_2}\cdots\mathbf{A}_{x_t}\boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty \quad . \tag{1}$$

Let x be a sequence of observations produced by a dynamical system, and $f_\mathcal{A}(x)$ be the probability that the system produces x. The vector $\boldsymbol{\alpha}_\lambda$ represents the initial state of the system.

[1]We assume for simplicity that all options are available from all states; otherwise the maps Δ_ω have domain I_ω instead of S and some non-essential technical issues arise when defining Δ.

If a history $u \in \Sigma^\star$ has already been observed, the conditional probability of observing a sequence of observations $v \in \Sigma^\star$ after u is:

$$f_{\mathcal{A},u}(v) = \frac{f_{\mathcal{A}}(uv)}{f_{\mathcal{A}}(u)} = \frac{\boldsymbol{\alpha}_\lambda^\top \mathbf{A}_u \mathbf{A}_v \boldsymbol{\alpha}_\infty}{\boldsymbol{\alpha}_\lambda^\top \mathbf{A}_u \boldsymbol{\alpha}_\infty} = \frac{\boldsymbol{\alpha}_u^\top \mathbf{A}_v \boldsymbol{\alpha}_\infty}{\boldsymbol{\alpha}_u^\top \boldsymbol{\alpha}_\infty} \quad . \tag{2}$$

Hence, given some history u, the initial state $\boldsymbol{\alpha}_\lambda$ can be updated to a new state $\boldsymbol{\alpha}_u / (\boldsymbol{\alpha}_u^\top \boldsymbol{\alpha}_\infty)$, which allows computing probabilities of future observations conditioned on the current history. Because the partition of a sequence of observations x into a history u and a future v (also called *test*) yields $x = uv$, we sometimes call histories *prefixes* and futures *suffixes*.

Theorem 1. *Let M be an MDP with n states, Ω a set of options, and $\Sigma = \Omega \times \{\sharp, \perp\}$. 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). Note that the MDP transition kernel and the options' stochastic policies are coupled inside the transition matrices of ODM $\mathcal{A}$. This coupling is the reason why we can model the timing of options in an MDP via a process with observation on the set $\Omega \times \{\sharp, \perp\}$ whose size is independent of set of actions A, and whose transitions operators have size at most $|S|$. Note that this can be particularly interesting in settings where the number of possible actions is very large but a small number of options is enough to specify the "useful" modes of operation of an agent.

An ODM $\mathcal{A}$ can be used to query the probability of observing any trajectory in $\Sigma^\star$ starting from a state sampled from $\boldsymbol{\alpha}$. In principle, this includes trajectories which are not valid for an agent interacting with an MDP via options – e.g. we can query the probability of trajectories of the form $(\omega_1, \sharp)^{d_1} (\omega_2, \sharp)^{d_2} \ldots$, where ω_1 was interrupted before termination. Note that this type of trajectories will never be observed by an agent that explores an environment by interacting with it only via options executed in call-and-return fashion. In particular, the agent would not need to learn about these trajectories if the goal is to plan via options only. We now show that an ODM can be restricted to produce non-zero probabilities for legal trajectories only, without increasing the size of the model.

If options are always executed to termination, valid trajectories always belong to a subset of $\Sigma^\star$, denoted V. The probability of asequence of options $\bar{\omega} = \omega_1 \cdots \omega_t$ and their durations $\bar{d} = d_1 \cdots d_t$, $d_i > 0$. is given by:

$$\mathbb{P}[\bar{d} | \alpha, \bar{\omega}] = \boldsymbol{\alpha}^\top \mathbf{A}_{\omega_1,\sharp}^{d_1 - 1} \mathbf{A}_{\omega_1,\perp} \mathbf{A}_{\omega_2,\sharp}^{d_2 - 1} \mathbf{A}_{\omega_2,\perp} \cdots \mathbf{A}_{\omega_t,\sharp}^{d_t - 1} \mathbf{A}_{\omega_t,\perp} \mathbf{1} \quad . \tag{3}$$

This can be computed using the ODM $\mathcal{A} = \langle \boldsymbol{\alpha}, \mathbf{1}, \{\mathbf{A}_\sigma\}\rangle$ from Theorem 1. Now we want to modify this model so that it only assigns non-zero probabilities to valid trajectories. That is, if f is the function computed by $\mathcal{A}$, we want another PSR that computes the function:

$$\tilde{f}(x) = \begin{cases} f(x) & x \in V \ , \\ 0 & x \notin V \ . \end{cases} \tag{4}$$

The following result shows that $\tilde{f}$ can also be computed by a PSR with almost the same size as $\mathcal{A}$.

Theorem 2. *If $\mathcal{A}$ has n states, then there exists a PSR with at most $(|\Omega| + 1)n$ states computing $\tilde{f}$.*

3 Spectral Learning of ODM

Because an option duration model over valid trajectories can be represented with a PSR of moderate size, we can use the spectral learning algorithm in [1] to estimate an ODM from a set of sampled trajectories $\Sigma^\star$ produced by the agent. For each trajectory the initial state is sampled according to a fixed distribution α. We assume that the options executed by the agent are selected according to some fixed open-loop policy. This is important if we want to use the sampled trajectories to learn a model of the environment which is independent of the exploration policy.

The algorithm takes as input Σ and a basis $\mathcal{B}$ in $\Sigma^\star$, uses them to estimate the corresponding Hankel matrices. The Hankel matrix is a convenient algebraic way to summarize a dynamical system. It is a bi-infinite matrix, $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$ with rows and columns indexed by strings in $\Sigma^\star$, which contains the joint probabilities of prefixes and suffixes. In many situations of interest, including POMDPs, the Hankel matrix has finite rank. Instead of looking at the full Hankel matrix, learning algorithms (including ours) usually work with finite sub-blocks of this matrix. Once we have the Hankel matrices, we can then recover a PSR by performing singular value decomposition and linear algebra operations on these matrices. Although these methods work almost out-of-the-box, in practice the results tend to be sensitive to the choice of basis. We now give a procedure for building a basis which is tailored for the case of learning option duration models.

Our procedure is parametrized by the maximum possible duration T_ω of each option; an upper bound $K_r \geq 1$ on the number of option executions needed to reach every possible state in M when initial states are sampled from α; and, an upper bound $K_d \geq 1$ on the number of option executions needed to distinguish every pair of states in M in terms of option duration information. These quantities can be given or obtained via cross-validation. The intuition is simple: we want to ensure that we have enough prefixes in the Hankel matrix to get to all reachable states in M, and enough suffixes to make sure each of these states can be distinguished from each other. The following construction formalizes this intuition.

We obtain the set $\mathcal{P}$ of prefixes in the basis as the union of two disjoint sets. The first set is denoted by $\mathcal{P}_\perp$ and is defined as follows:

$$\mathcal{P}_\perp = \left\{ x_1 x_2 \cdots x_t \;\middle|\; 1 \leq t \leq K_r, x_i = (\omega_i, \sharp)^{d_i} (\omega_i, \perp), \omega_i \in \Omega, 0 \leq d_i \leq T_{\omega_i} \right\} \ . \tag{5}$$

These are trajectories with at most K_r option executions, containing all possible sequences of options, and all possible option durations allowed by the model.Note that each trajectory in $\mathcal{P}_\perp$ terminates with an option termination symbol of the form $(\omega, \perp)$. If we remove this last symbol for each possible trajectory, we obtain a disjoint set of prefixes: $\mathcal{P}_\sharp = \{x \mid x(\omega, \perp) \in \mathcal{P}_\perp\}$. Then we take $\mathcal{P} = \mathcal{P}_\perp \cup \mathcal{P}_\sharp$.

Similarly, the set of suffixes will be obtained as the union of two sets. These are defined as follows:

$$\mathcal{S}_\sharp = \left\{ x_1 x_2 \cdots x_t \;\middle|\; 1 \leq t \leq K_s, x_i = (\omega_i, \sharp)^{d_i} (\omega_i, \perp), \omega_i \in \Omega, 0 \leq d_i \leq T_{\omega_i} \right\} \ , \tag{6}$$

$$\mathcal{S}_\perp = \{(\omega, \perp) x \mid x \in \mathcal{S}_\sharp, \omega \in \Omega\} \ . \tag{7}$$

The suffixes in $\mathcal{S}_\sharp$ are obtained like the prefixes in $\mathcal{P}_\perp$, with the only difference that the number of option executions is now upper bounded by K_s instead of K_r. Suffixes in $\mathcal{S}_\perp$ are obtained by prefixing each string in $\mathcal{S}_\sharp$ with each possible option termination symbol $(\omega, \perp)$. The whole set of suffixes is $\mathcal{S} = \mathcal{S}_\perp \cup \mathcal{S}_\sharp$

Note that not every string in $\mathcal{PS}$ defines a valid trajectory. This is required for the Hankel matrices $\mathbf{H}_\sigma$ to be different from zero; otherwise the operators $\mathbf{A}_\sigma$ cannot be correctly recovered. To see why this is the case, consider the basis $\mathcal{B}' = (\mathcal{P}_\perp, \mathcal{S}_\sharp)$. By construction we have $\mathcal{P}_\perp \mathcal{S}_\sharp \subset V$. However, if we consider a system where some ω never lasts for just one step, then every trajectory in $\mathcal{P}_\perp\{(\omega, \perp)\}\mathcal{S}_\sharp$ has probability zero. In particular, in such a system the matrix $\mathbf{H}_\sigma$ over the basis $\mathcal{B}'$ is exactly zero. To work around this problem, it is necessary to introduce non-valid trajectories in the basis, to ensure that $\mathbf{H}$ will contain some sub-blocks filled with zeros, but the $\mathbf{H}_\sigma$ will contain some non-zero sub-blocks.

4 Experimental Results

We first illustrate our approach on synthetic grids of different sizes. We use 4-connected grids 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 episodes in which ten options are executed up to termination. We used grids of sizes $d \times d$ with $d \in \{5, 9, 13\}$. For each grid we also considered two possible initial conditions: one under which the initial state is sampled uniformly at random , and the other where the agent deterministically starts from the center of the grid (reported here).

We sampled training sets of different sizes and for each, we learned an ODM as described above. To evaluate these ODMs we consider newly sampled trajectories and for each of them compare the duration of options in the trajectory with the expected durations predicted by the learned model

Figure 1: Left: relative error vs. sample size for different grid size and history length 2. Middle: comparision against a naive prediction strategy using only the empirical mean durations with $d = 5$ and history length 2. Right: relative error vs. history length for $d = 13$. In all cases we choose $K_r = K_s = 2$.

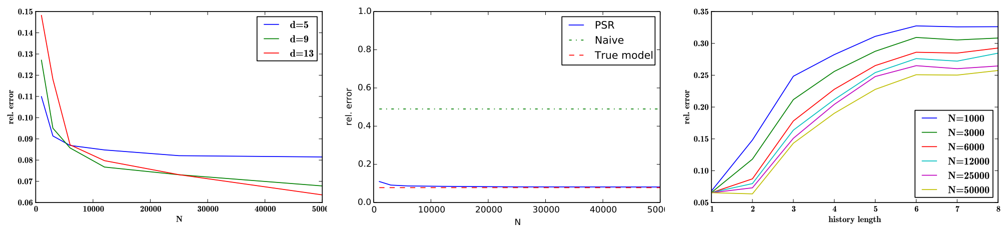

Table 1: Relative accuracy and optimal number of states for different grid sizes d, basis parameters (K_r, K_s), and history length h. Training sample size fixed to $N = 50000$.

d	(K_r, K_s)	$h = 1$	$h = 2$	$h = 3$	$h = 4$	$h = 5$	$h = 6$	$h = 7$	$h = 8$
5	(1 , 1)	0.10 (29)	0.19 (24)	0.25 (19)	0.28 (16)	0.30 (12)	0.30 (14)	0.31 (12)	0.31 (14)
	(1 , 2)	0.10 (29)	0.20 (26)	0.25 (29)	0.28 (29)	0.30 (29)	0.30 (19)	0.31 (19)	0.31 (19)
	(2 , 1)	0.10 (29)	0.10 (24)	0.14 (23)	0.17 (23)	0.19 (23)	0.20 (23)	0.21 (23)	0.21 (23)
	(2 , 2)	0.09 (25)	0.08 (28)	0.13 (28)	0.16 (28)	0.19 (23)	0.19 (23)	0.20 (23)	0.20 (23)
9	(1 , 1)	0.16 (49)	0.29 (44)	0.37 (53)	0.41 (54)	0.43 (32)	0.44 (32)	0.44 (32)	0.44 (32)
	(1 , 2)	0.18 (62)	0.31 (61)	0.38 (34)	0.42 (41)	0.44 (36)	0.44 (37)	0.45 (37)	0.46 (40)
	(2 , 1)	0.08 (42)	0.21 (87)	0.24 (39)	0.25 (41)	0.26 (38)	0.26 (38)	0.26 (38)	0.25 (40)
	(2 , 2)	0.08 (57)	0.07 (60)	0.14 (71)	0.19 (59)	0.22 (59)	0.24 (59)	0.25 (59)	0.25 (59)
13	(1 , 1)	0.16 (49)	0.30 (49)	0.40 (49)	0.44 (49)	0.48 (78)	0.48 (53)	0.48 (51)	0.48 (53)
	(1 , 2)	0.20 (77)	0.37 (56)	0.43 (75)	0.47 (80)	0.49 (79)	0.50 (75)	0.50 (75)	0.51 (76)
	(2 , 1)	0.07 (51)	0.21 (149)	0.25 (146)	0.26 (172)	0.27 (87)	0.26 (48)	0.25 (46)	0.26 (46)
	(2 , 2)	0.07 (70)	0.06 (91)	0.14 (91)	0.19 (91)	0.23 (91)	0.25 (92)	0.25 (91)	0.26 (91)

given the current history. We do this for all options in a trajectory, and report the relative deviation between observed and predicted duration as a function of the length of the observed history.

In each case, prediction accuracies are reported on a validation set with 2000 trajectories. A test set of 2000 samples was used to select the best number of states $1 \leq n \leq d^2$ and basis parameters $K_r, K_s \in \{1, 2\}$. The maximum durations of options needed to build the basis are estimated from the training data. Results are reported in Table 1 and Figure 1. The results show that larger basis and larger samples yield better models, and that predicted durations degrade as the length of the history increases. This last finding is an effect of using a fixed initial state for the exploration; if the initial state is chosen uniformly at random the prediction accuracy is more consistent across history lengths. Note also how for smaller basis the best number of states to predict optimally with different history lengths is more unstable.

We also tested the approach in a simulated robot with continuous state and nonlinear dynamics, in which the actions correspond to actuators. We use the Box2D physics engine[2] to obtain realistic accelerations, collisions and friction effects for a circular differential wheeled robot. We set the angular and linear damping to 0.9 and 0.99 respectively and use a coefficient of restitution of 0.1. A primitive action in this domain consists in the application of force vector of $(0.5, 0.5)$ on the wheels every $1/10$ of a simulated second.

Like in the synthetic case, we define a set of options; each option is identified by a radial direction and is terminated when the robot hits a wall. We used sets of 4 and 8 options, describing directions uniformly distributed across the unit circle. The environment simulates an empty square world of size 4×4 units. Models where trained with 15000 trajectories and the number of states is selected by searching in $1 \leq n \leq 200$ with a test set containing 2000 trajectories. Accuracies for different basis sizes are reported in Table 2.

[2]`http://box2d.org/`

Table 2: Relative accuracies and optimal number of states across several basis and history lengths for the robot simulator with 4 and 8 options.

$\|\Omega\|$	(K_r, K_s)	$h = 1$	$h = 2$	$h = 3$	$h = 4$	$h = 5$	$h = 6$	$h = 7$	$h = 8$
4	(2 , 1)	0.19 (199)	0.25 (199)	0.26 (196)	0.30 (198)	0.31 (172)	0.33 (163)	0.31 (173)	0.30 (172)
	(1 , 1)	0.15 (133)	0.28 (126)	0.31 (134)	0.35 (131)	0.36 (131)	0.36 (131)	0.36 (132)	0.36 (133)
8	(2 , 1)	0.40 (176)	0.47 (163)	0.49 (163)	0.51 (176)	0.52 (162)	0.51 (164)	0.50 (163)	0.52 (167)
	(1 , 1)	0.38 (166)	0.48 (162)	0.46 (195)	0.51 (164)	0.52 (162)	0.51 (162)	0.51 (165)	0.54 (169)

5 Discussion

The approach we presented learns a predictive model for option durations. Using similar techniques it is also possible to show that a value function for an MDP can actually be expressed as a linear combination of the predictive state features obtained in this model. We are currently experimenting with planning using such models. We note that modelling states in terms of what happens after executing certain actions is known to be useful, e.g. [4]. But our models are simplified, in order to avoid handling large observation spaces. In this respect, our work differs significantly both from action-respecting embeddings [2, 3] and predictive state representations with options [15], which aim to learn full observation models conditioned on extended actions, which characterize the current state. Timing models get around the problem of both large action spaces (by using a finite set of options) and the problem of large observation spaces (by focusing only on continuation and termination). In this last respect it is worth mentioning that we have observed Theorem 2 not to be tight in the grid MDP considered in our synthetic experiments: there the number of states of the best ODM grows like $O(\sqrt{n})$ instead of $O(n)$. This suggests it might be possible to show that an ODM can be learned faster than the corresponding underlying MDP.

References

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

[2] M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. In *ICML*, 2005.

[3] M. Bowling, D. Wilkinson, A. Ghodsi, and A. Milstein. Subjective localization with action respecting embedding. *Robotics Research*, 2007.

[4] P. Dayan. Improving generalisation for temporal difference learning: The successor representation. *Neural Computation*, 1993.

[5] W. Droste, M. Kuich and H. Vogler. *Handbook of weighted automata*. Springer, 2009.

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

[7] H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 2000.

[8] M. Littman, R. Sutton, and S. Singh. Predictive representations of state. In *NIPS*, 2002.

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

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

[11] M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. In *ICML*, 2004.

[12] S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *UAI*, 2004.

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

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

[15] B. Wolfe and S. Singh. Predictive state representations with options. In *ICML*, 2006.