Linear Options

Jonathan Sorg
Computer Science & Engineering
University of Michigan
jdsorg@umich.edu

Satinder Singh
Computer Science & Engineering
University of Michigan
baveja@umich.edu

ABSTRACT

Learning, planning, and representing knowledge in large state spaces at multiple levels of temporal abstraction are key, long-standing challenges for building flexible autonomous agents. The options framework provides a formal mechanism for specifying and learning temporally-extended skills. Although past work has demonstrated the benefit of acting according to options in continuous state spaces, one of the central advantages of temporal abstraction—the ability to plan using a temporally abstract model—remains a challenging problem when the number of environment states is large or infinite. In this work, we develop a knowledge construct, the linear option, which is capable of modeling temporally abstract dynamics in continuous state spaces. We show that planning with a linear expectation model of an option's dynamics converges to a fixed point with low Temporal Difference (TD) error. Next, building on recent work on linear feature selection, we show conditions under which a linear feature set is sufficient for accurately representing the value function of an option policy. We extend this result to show conditions under which multiple options may be repeatedly composed to create new options with accurate linear models. Finally, we demonstrate linear option learning and planning algorithms in a simulated robot environment.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Reinforcement Learning, Temporal Abstraction

1. INTRODUCTION

In the field of Reinforcement Learning (RL), agents are placed in unknown environments and have to learn how to act so as to maximize rewards over a horizon. In complex environments, it is critical that an agent be able to handle knowledge at multiple levels of abstraction. We distinguish between two types of abstraction – state abstraction and temporal abstraction.

State abstraction is closely related to the concept of *generalization*, whereby knowledge from one situation can be used and applied in other distinct but related situations. In addition to drastically reducing the data complexity of learning, state abstraction can reduce computational burden by allowing planning to compute a good policy for many situations at once. In environments with continuously valued observations or others with an large number of states, some form of generalization is necessary.

Linear methods are perhaps the most common and best understood class of generalization mechanisms. Linear methods encompass a wide spectrum of approaches including look-up tables, state aggregation methods, radial basis functions with fixed bases, and others. They have a long history in the field of RL and form the basis of many methods with strong theoretical guarantees.

Temporal abstraction also provides both sample complexity and computational efficiency benefits. A model that is incapable of temporal abstraction accounts for distal events in time through repeated composition of short-term effects. In contrast, an agent equipped with abstract temporal reasoning can account for temporally distal effects of action in a single atomic step. This can drastically reduce the computational burden of planning for long-term goals, and it also can improve a model's accuracy. Errors propagate and compound through repeated composition, but by representing distant effects directly, a model can reduce error in its prediction. Because temporal effects are inexorably linked to an agent's decisions, temporal abstraction mechanisms are tightly coupled with control.

Based on the theory of Semi-Markov Decision Processes (SMDPs), the *options* framework [10] provides a formal method for representing temporally abstract knowledge. In addition to providing a unified framework for specifying subgoals and skills, options have an associated model representation which captures the joint distribution over termination states and time. Methods for MDP planning with models based on primitive actions can thus be easily extended to SMDP planning with models based on options.

The options framework does not explicitly provide a mechanism for state abstraction. However, the notion of a skill often refers to abstraction of both kinds. For skill learning to be most useful, a skill ought to be reusable in many contexts. For example, once an agent learns the skill of opening a door, ideally that agent would then have mastered the ability to open many doors. This requires the ability to generalize the

Cite as: Linear Options, Jonathan Sorg and Satinder Singh, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

skill across doors.

There have been a number of approaches combining state and temporal extraction. In MAXQ [4], the two are coupled in a hierarchical decomposition of the state space. Konidaris and Barto [5] define an explicit abstract state space which is shared across related environments. The relational RL community has combined options with relational abstractions [3]. Most closely related to our work are applications of options to continuous state spaces [6, 9]. These works tackle the problems of representing policies at different levels of abstraction. However, the ability to *plan* using a temporally abstract model—one of the central advantages of temporal abstraction—remains a challenging problem when the number of states is large.

Our main contributions in this paper are a knowledge structure called the *linear option* which provides a general mechanism for providing both state and temporal abstraction capabilities, proofs that learning and planning with linear options and their models produces sensible behavior, conditions on feature representations under which compositions of linear option models are accurate, and finally a simple empirical verification of the feasibility and advantages of linear options.

2. BACKGROUND

An MDP is a tuple $\langle S, A, P, R, \gamma \rangle$ consisting of a state set S, action set A, transition function $P : S \times A \times S \to [0, 1]$, an expected reward function $R : S \times A \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. Every MDP time step t, the agent takes an action a_t in state s_t, and the environment responds with a reward r_t and resulting next state $s'_t = s_{t+1}$. A policy $\pi : S \to A$ is a mapping from states to actions. An action-value function or Q-function, $Q^\pi(s, a)$ is the expected discounted reward obtained by taking action a in state s and following π thereafter. The Bellman equation defining the optimal Q-function is: $\forall s \in S, a \in A$,

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \max_{b \in A} Q^*(s', b). \quad (1)$$

A policy that is greedy with respect to a given Q-function maps each state to $\arg\max_a Q(s, a)$. The value function $V(s)$ corresponding to a given Q-function is the value of the greedy action from each state: $V(s) = \max_a Q(s, a)$. A policy that is greedy with respect to the optimal value function is an optimal policy.

2.1 Linear Methods for MDPs

In the reinforcement learning literature, linear methods have long been used for value function approximation. Instead of using the state as an index into a table, general linear methods map each state s to a corresponding n-dimensional feature vector $\phi(s) \in \mathbb{R}^n$. The value of a policy π is approximated using $V^\pi(s) \approx \phi(s)^\mathsf{T}\theta^\pi$, where θ^π is the parameter vector to be learned.

Linear Temporal Difference (TD) learning evaluates a policy π through repeated applications of an incremental gradient-descent-like update rule. A t-length sequence of experience is in the form $(\phi_1, a_1, r_1, \phi_2, a_2, r_2, \ldots, \phi_t, a_t, r_t, \phi_{t+1})$. We translate this into a sequence of start-state feature vectors $(\phi_1, \phi_2, \ldots, \phi_t)$, a corresponding sequence of resulting-state feature vectors $(\phi'_1, \phi'_2, \ldots, \phi'_t) = (\phi_2, \phi_3, \ldots, \phi_{t+1})$, and a sequence of rewards $(r_1, r_2, \ldots, r_t)$, all generated by following policy π. TD learning updates the value function parameters θ^π at each time step according to

$$\theta^\pi_{t+1} = \theta^\pi_t + \alpha_t \phi_t (r_t + \gamma \phi_t'^\mathsf{T} \theta^\pi_t - \phi_t^\mathsf{T} \theta^\pi_t), \quad (2)$$

where α_t is a schedule of learning-rate step sizes. Linear TD learning has been shown [13] to converge to an approximation of V^π, assuming a suitably decreasing schedule of step sizes (α_t) for the incremental updates in Equation (2).

Least-Squares TD: Linear Least-Squares Temporal Difference learning for Markov decision processes [2, 1]—which we denote MDP-LSTD—achieves a lower sample complexity than TD learning by exploiting the following observation: given a fixed setting of the value function parameters θ^π, we can explicitly represent the sum of TD updates that would have occurred to θ^π using all of the experience up to time t as

$$u_t = \alpha \sum_{k=1}^{t} \phi_k (r_k + \gamma (\phi'_k)^\mathsf{T} \theta^\pi - \phi_k^\mathsf{T} \theta^\pi).$$

MDP-LSTD estimates θ^π as the value for which u_t is 0. Algebraically, this is

$$\theta^\pi = \left[\sum_{k=1}^{t} \phi_k (\phi_k - \gamma \phi'_k)^\mathsf{T} \right]^{-1} \left[\sum_{k=1}^{t} \phi_k r_k \right].$$

It is known that MDP-LSTD and standard TD learning converge to the same fixed point [1].

2.2 Options and SMDPs

An option can be thought of as a subroutine or skill that an agent can invoke in certain states. Formally, an option, o, in an MDP is defined by the tuple $\langle \mathcal{I}_o, \mu_o, \beta_o \rangle$. The set $\mathcal{I}_o \subseteq S$ is the set of states in which the option can be invoked, the option policy $\mu_o : S \to A$ maps each state to an action, and $\beta_o : S \to [0, 1]$ specifies the probability of termination in each state s. A primitive action $a \in A$ can be thought of as a special case of an option that terminates with probability one in every state and so lasts exactly one time step, and is available everywhere action a is available. A set of options $\mathcal{O}$ can thus contain multi-step options as well as primitive actions defined as one-step options.

MDP + Options = SMDP. An MDP on which options are defined can be viewed as a Semi-Markov Decision Process (SMDP). In short, an SMDP is quite like an MDP, except we allow options to take varying amounts of time, depending on state. Each *SMDP time step*, the agent selects an option and follows that option's policy until termination. The option is run until it terminates, lasting a random number of *MDP time steps*, which we refer to as the option's duration d. Unless otherwise specified, "time step" hereafter refers to an SMDP time step.

More precisely, an SMDP consists of a state space S, an option set $\mathcal{O}$, a discount parameter, $\gamma \in [0, 1)$, and for each state and option, a reward function $R(s, o)$, defined as the expected cumulative discounted reward obtained until the option terminates, and a joint distribution $P(s', d|s, o)$ over termination state s' and duration d.

Option Models. An option model must specify the distribution over option duration in addition to the usual distribution over termination states. Fortunately, for the purposes of planning, we can use a simple trick to fold the duration neatly within the typical transition model parametriza-

tion. Define

$$P(s'|s,o) = \sum_{d=1}^{\infty} \gamma^d P(s',d|s,o),$$

where $P(s',d|s,o)$ is the joint distribution over termination state s' and option duration d. We will refer to $P(s'|s,o)$ as option o's transition model. An option's reward model computes the expected discounted reward obtained during the execution of the option:

$$R(s,o) = \mathbb{E}\left\{ r_t + \cdots + \gamma^{d-1} r_{t+d-1} | s_t = s, o_t = o \right\},$$

where subscripts denote time and d is the duration of the option. Using these definitions, the Bellman equation for the SMDP can be expressed succinctly as follows:

$$\begin{aligned} Q^*(s,o) &= \mathbb{E}\left\{ r_t + \cdots + \gamma^{d-1} r_{t+d-1} + \gamma^d V^*(s_{t+d}) \right\} \\ &= R(s,o) + \sum_{s' \in S} P(s'|s,o) \max_{o' \in \mathcal{O}} Q^*(s',o'). \end{aligned}$$

Behavior Policy: When choosing how to act, an agent will select among the options whose initiation set includes the current state. We will refer to the policy followed by the agent, π, as the *behavior policy*, where $\pi : S \to \mathcal{O}$ denotes that the agent selects option o in state s if the previously executing option has just terminated. Instead of maintaining Q-function estimates for each action, a learning/planning agent will maintain estimates of the Q-function $Q^*(s,o)$, for each option $o \in \mathcal{O}$. We will refer to these estimates as the *behavior value function*.

3. LINEAR OPTIONS

As the name suggests, a linear option is a direct extension of the options framework from the tabular representation to the more general linear representation. The basic definition of an option remains the same; however, the quantities are defined over the n-dimensional feature space instead of being defined directly over states. A linear option is a tuple $o = \langle \mathcal{I}_o, \mu_o, \beta_o \rangle$, where $\mathcal{I}_o \subseteq \mathbb{R}^n$ is the initiation set, $\mu_o : \mathbb{R}^n \to A$ is the option policy, and $\beta_o : \mathbb{R}^n \to [0,1]$ is the probability of termination given a feature vector. To simplify the theorems that follow, we consider options which are available in all states ($\mathcal{I}_o = \mathbb{R}^n$).

Note that since linear options use feature based representations of state, they apply just as well to continuous state MDPs as to discrete state MDPs (we do, however, need the discrete time assumption). Indeed, most of the discussion that follows applies equally to discrete and continuous state MDPs. In particular, our empirical result will involve a continuous domain.

3.1 Evaluation of a Behavior Policy

Although our eventual goal is to present the form of a linear option model, in order to prove that planning with such a model converges to a good result, it is useful to provide a good basis for comparison. It was recently shown by both Parr et al. [7] and Sutton et al. [12] that policy evaluation with a primitive-action linear expectation model converges to the the same solution as does MDP-LSTD. Because MDP-LSTD is only defined for primitive actions, in this section, we define SMDP Least-Squares Temporal Difference learning (SMDP-LSTD), a straightforward extension of MDP-LSTD to the SMDP setting.

The development of SMDP-LSTD is similar to the development of MDP-LSTD. While following behavior policy π, if an option is initiated at time t, the resulting experience is in the form of a 4-tuple $\langle \phi_t, d_t, r_t, \phi'_t \rangle$, where ϕ_t and ϕ'_t are the feature representations of the start and termination states, r_t is the discounted sum of rewards from start until termination, and d_t is the option's duration. We also define the effective discount factor $\gamma_t = \gamma^{d_t}$.

The sole difference between SMDP-LSTD and its primitive-action counterpart is the fact that the effective discount factor is different for each observed transition. Otherwise, we can define the sum of TD updates and set this equation to 0 as was done for a primitive-action policy in standard LSTD. This gives the solution

$$0 = \sum_{k=1}^{t} \phi_k (r_k + \gamma_k (\phi'_k)^\mathsf{T} \theta^\pi - \phi_k{}^\mathsf{T} \theta^\pi) \tag{3}$$

$$\theta^\pi = \left[\sum_{k=1}^{t} \phi_k (\phi_k - \gamma_k \phi'_k)^\mathsf{T} \right]^{-1} \left[\sum_{k=1}^{t} \phi_k r_k \right]. \tag{4}$$

We will refer to these equations in the next section, which develops the linear-option expectation model.

3.2 Linear Model of a Behavior Policy

A linear-option expectation model of a behavior policy's dynamics, denoted ($\mathbf{F}_\pi$,$\mathbf{b}_\pi$) will attempt to satisfy, for all time steps t,

$$\begin{aligned} \mathbf{F}_\pi \phi_t &\approx \mathbb{E}[\gamma_t \phi'_t | \phi_t] \\ \mathbf{b}_\pi^\mathsf{T} \phi_t &\approx \mathbb{E}[r_t | \phi_t]. \end{aligned}$$

A learning and planning agent will need to estimate $\mathbf{F}_\pi$ and $\mathbf{b}_\pi$ from data. While following policy π, at SMDP time step t, the agent will have gathered a dataset of experience as a set of tuples of the above form. To more compactly represent the theorems that follow, we will represent this dataset in the form of matrices. Let $\mathbf{\Phi}_t$ be a matrix representing the sequence of starting feature vectors, let matrix $\mathbf{\Phi}'_t$ represent the sequence of termination-state feature vectors, let $\mathbf{r}_t$ be a vector of observed SMDP rewards, and let $\mathbf{\Gamma}_t$ be a vector of effective discount factors:

$$\mathbf{\Phi}_t = \begin{bmatrix} \phi_1{}^\mathsf{T} \\ \phi_2{}^\mathsf{T} \\ \vdots \\ \phi_t{}^\mathsf{T} \end{bmatrix}, \mathbf{\Phi}'_t = \begin{bmatrix} \phi'_1{}^\mathsf{T} \\ \phi'_2{}^\mathsf{T} \\ \vdots \\ \phi'_t{}^\mathsf{T} \end{bmatrix}, \mathbf{r}_t = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_t \end{bmatrix}, \mathbf{\Gamma}_t = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_t \end{bmatrix}.$$

We will drop the subscript t from the notation of these matrices hereafter. In Definition 1, we present the least-squares solution of the model parameters given this dataset.

DEFINITION 1. *Given a dataset* $(\mathbf{\Phi}, \mathbf{\Phi}', \mathbf{\Gamma}, \mathbf{r})$ *generated by following behavior policy* π*, the linear-option expectation-model (LOEM) for behavior policy* π*, denoted* $(\mathbf{F}_\pi, \mathbf{b}_\pi)$*, is given by*

$$\mathbf{F}_\pi{}^\mathsf{T} = (\mathbf{\Phi}'\mathbf{\Phi})^{-1} \mathbf{\Phi}^\mathsf{T} \operatorname{diag}(\mathbf{\Gamma}) \mathbf{\Phi}' \tag{5}$$

$$\mathbf{b}_\pi = (\mathbf{\Phi}'\mathbf{\Phi})^{-1} \mathbf{\Phi}^\mathsf{T} \mathbf{r}, \tag{6}$$

where $\operatorname{diag}(\mathbf{\Gamma})$ is the diagonal matrix with the elements of $\mathbf{\Gamma}$ ordered along the diagonal. Notice that the discount factors

fold into the transition expectation model $\mathbf{F}_\pi$ just as they did in the discrete definition of an SMDP transition model.

This expectation model can be used for behavior policy evaluation. Given an initial feature vector ϕ, we can compute the expected discounted termination feature vector $\mathbf{F}_\pi\phi$ as well as the expected discounted reward until termination $\mathbf{b}_\pi^\mathsf{T}\phi$. Using the values of this expected experience, we can improve the estimate of the value function parameters using a TD update. We will refer to this operation as a LOEM policy evaluation update.

Given an input feature vector ϕ, the following recursion defines a LOEM policy evaluation update to the value function parameters θ_k:

$$\theta_{k+1} = \theta_k + \alpha_k(\mathbf{b}_\pi^\mathsf{T}\phi + \theta_k^\mathsf{T}\mathbf{F}_\pi\phi - \theta_k^\mathsf{T}\phi)\phi, \tag{7}$$

for some step size α_k.

Notice that the expected discounted termination feature vector, $\mathbf{F}_\pi\phi$, may not correspond to the feature representation of any actual state in the MDP. Nevertheless, we show that if the agent is committed to representing its value function linearly, value updates of the form in Equation (7) will converge to a useful value.

The following theorem shows that LOEM policy evaluation updates in Equation (7) converge to the same behavior value function parameters as does the SMDP-LSTD algorithm. In the theorem, we will require a bound on the numerical radius of a matrix. This is $r(\mathbf{A}) = \max_{\|x\|_2=1} x^\mathsf{T}\mathbf{A}x$, for some matrix $\mathbf{A}$. Essentially, this bound excludes expectation models which predict that future feature vectors will grow without bound. For a more detailed discussion of this, see Sutton et al. [12].

THEOREM 1. *Consider the TD iteration using the linear-option expectation model for behavior policy π with a non-negative step-size sequence (α_k):*

$$\theta_{k+1} = \theta_k + \alpha_k(\mathbf{b}_\pi^\mathsf{T}\phi_k + \theta_k^\mathsf{T}\mathbf{F}_\pi\phi_k - \theta_k^\mathsf{T}\phi_k)\phi_k \tag{8}$$

where $\theta_0 \in \mathbb{R}^d$ is arbitrary. Assume that (i) the step-size sequence satisfies $\sum_{k=0}^\infty \alpha_k = \infty$,$\sum_{k=0}^\infty \alpha_k^2 < \infty$, (ii) $r(F) < 1$, (iii) (ϕ_k) are uniformly bounded i.i.d. random variables, and that (iv) $\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi}$ is non-singular. Then the parameter vector θ_k converges with probability one to

$$\theta^\pi = (\mathbf{I} - \mathbf{F}_\pi^\mathsf{T})^{-1}\mathbf{b}_\pi, \tag{9}$$

which is the same solution as that returned by SMDP-LSTD.

PROOF. The proof of value function parameter convergence to the solution in Equation (9) using LOEM evaluation updates is a straightforward adaptation of the corresponding theorem in Sutton et al. [12], which we do not present here.

Here, we show that the LOEM solution (9) is equivalent to the SMDP-LSTD solution (4). We can re-express Equation (9) as

$$0 = \mathbf{b}_\pi + (\mathbf{F}_\pi{}^\mathsf{T} - \mathbf{I})\theta^\pi. \tag{10}$$

Define $\mathbf{B} = \sum_{k=1}^t \gamma_k\phi_k(\phi_k')^\mathsf{T} = \mathbf{\Phi}^\mathsf{T}\,\mathrm{diag}(\mathbf{\Gamma})\mathbf{\Phi}'$, define $\mathbf{C} = \mathbf{\Phi}^\mathsf{T}\mathbf{\Phi}$, and define $\bar{\mathbf{r}} = \sum_{k=1}^t \phi_k r_k = \mathbf{\Phi}^\mathsf{T}\mathbf{r}$. Given these definitions, Equation (3) is equivalent to

$$0 = \bar{\mathbf{r}} + (\mathbf{B} - \mathbf{C})\theta^\pi. \tag{11}$$

Equation (11) is equivalent to Equation (10) after multiplying both sides by $\mathbf{C}^{-1}$. Therefore any solution of (11) is also a solution of (10). Because all of the above steps are reversible, the reverse statement holds as well. □

3.3 Control with Linear Models

The previous sections have focused on behavior policy evaluation. Next, we consider the control problem of finding a good behavior policy. To do this using model-based planning, we will need more than an expectation model of a single behavior policy. We propose learning a separate expectation model for each option. A linear-option expectation model for an option o is equivalent to a linear-option expectation model for a special "single-option" behavior policy—the policy that picks option o in every state. Thus, for any option $o \in \mathcal{O}$, the linear-option expectation model for that option, $(\mathbf{F}_o, \mathbf{b}_o)$, can be defined and computed via Definition 1 using samples of option o executed to completion.

We represent Q-functions with a separate parameter vector θ_o for every option o, such that $Q(s,o) = \theta_o^\mathsf{T}\phi(s)$. In our empirical results we will use two types of updates to the Q-functions. The first is a planning update. Given a linear-option expectation model for each option, it considers the consequences of taking option o in state ϕ and updates θ_o as follows:

$$\theta_o = \theta_o + \alpha(\mathbf{b}_o^\mathsf{T}\phi + \max_{o'} \theta_{o'}^\mathsf{T}\mathbf{F}_o\phi - \theta_o^\mathsf{T}\phi)\phi. \tag{12}$$

The second is a model-free intra-option value learning update defined in the next section. In particular, we will use two agents, one that only uses intra-option value learning updates, and a second that combines one step of intra-option value learning update with one step of the planning update at each time step. The latter agent, which updates a value function through both model-free learning and model-based planning, was inspired by the Dyna architecture [11].

3.4 Intra-Option Learning

Traditional SMDP learning algorithms only update models and value functions when options terminate. However, even while executing options to completion, the agent is constantly executing primitive actions every MDP time step. Here we develop more data-efficient algorithms by using information observed while the option is executing. These methods are called intra-option learning methods. In this section, unless otherwise noted, "time step" refers to an MDP time step. Accordingly, r_t in this section refers to the instantaneous reward received every MDP time step.

Learning the Behavioral Value Function. After executing primitive action a in state ϕ, *intra-option value learning* can update the value estimate of every option o for which $\mu_o(\phi) = a$. This can be much more efficient than execute-to-completion SMDP value learning because the agent does not need to wait until termination to update the value of an option and because multiple options are updated at every time step.

Formally, after experiencing a transition $\langle\phi_t, a_t, r_t, \phi_t'\rangle$, the values for all options o such that $\mu_o(\phi_t) = a_t$ are updated according to

$$\theta_o \leftarrow \theta_o + \alpha\left[r_t + \gamma U(\phi_t', o) - \theta_o{}^\mathsf{T}\phi_t\right]\phi_t, \tag{13}$$

where the function U is defined as

$$U(\phi, o) = (1 - \beta(\phi))\theta_o{}^\mathsf{T}\phi + \beta(\phi) \max_{o':\phi\in\mathcal{I}_{o'}} \theta_{o'}{}^\mathsf{T}\phi. \quad (14)$$

Intuitively the function U is the value of the next state taking into account whether or not the option terminates. Equations (13) and (14) are a straightforward extension of the tabular intra-option rules to the linear setting. We reproduce it here because this method is used in the empirical section.

Learning the Model. *Intra-option model learning*, applies a similar technique to accelerate the training of an option model. Recall the learning target for a linear-option expectation transition model

$$\mathbb{E}[\gamma^d \phi_{t+d} | \phi_t, o_t] \approx \mathbf{F}_o \phi_t,$$

where d is the random duration of the transition. Instead of waiting d time steps, the linear-option expectation transition model can be updated after one MDP time step. We can define the linear expectation model recursively as

$$\mathbf{F}_o\phi_t \approx \mathbb{E}\left[\gamma\beta(\phi_{t+1})\phi_{t+1} + \gamma(1 - \beta(\phi_{t+1}))\mathbf{F}_o\phi_{t+1}\right]. \quad (15)$$

In words, if the option terminates in state ϕ_{t+1}, then the model should predict termination in state ϕ_{t+1}. Otherwise, the model should equal the discounted termination of initiating the option in state ϕ_{t+1}. Using the learning target in Equation (15), we can compute a sample estimate of the squared error of our model. Given a transition from ϕ_t to ϕ_{t+1}, define the sample squared-error loss function

$$\mathcal{L}(\mathbf{F}_o) = \left\|\mathbf{F}_o\phi_t - \gamma\beta(\phi_{t+1})\phi_{t+1} - \gamma(1 - \beta(\phi_{t+1}))\mathbf{F}_o\phi_{t+1}\right\|_2.$$

Let $\eta = \phi_t - \gamma(1 - \beta(\phi_{t+1}))\phi_{t+1}$. The gradient of the sample error with respect to the model parameters is,

$$\nabla_{\mathbf{F}_o}\mathcal{L}(\mathbf{F}_o) = \nabla_{\mathbf{F}_o}\|\gamma\beta(\phi_{t+1})\phi_{t+1} - \mathbf{F}_o\eta\|_2 \quad (16)$$

$$= -2\left(\gamma\beta(\phi_{t+1})\phi_{t+1} - \mathbf{F}_o\eta\right)\eta^\mathsf{T}. \quad (17)$$

This leads to the intra-option model update rule:

$$\mathbf{F}_o \leftarrow \mathbf{F}_o + \alpha\left(\gamma\beta(\phi_{t+1})\phi_{t+1} - \mathbf{F}_o\eta\right)\eta^\mathsf{T}. \quad (18)$$

Much like the intra-option value learning rule, this can be applied to all options o for which $\mu_o(\phi_t) = a_t$. The intra-option learning rule for the reward model $\mathbf{b}_o$ can be defined similarly.

Option Policy Learning. Finally, we note that given a set of subgoal states, an option policy for reaching those subgoal states can be learned. The agent defines a *pseudo-reward function* specific to the subgoal states. An option designed to reach a doorway, for example, might assign a pseudo-reward of 1 for reaching the doorway and a small penalty elsewhere. The agent can then maintain a pseudo-Q-function estimating the value of the option's policy μ_o with respect to the pseudo-reward. The policy can then be learned via an off-policy learning method such as Q-learning. In our experiments, we represent the pseudo-Q-function using the same features as used for the behavioral value function.

4. OPTION COMPOSITION

Composing models of options is at the heart of planning and learning with options. In this section we provide conditions under which linear-option expectation models may be accurately composed. In the process, we extend recent results on linear feature selection in MDPs to the SMDP case.

4.1 On Feature Representations

Previously, we defined $\mathbf{\Phi}$ to be a time-growing data set of the observed states. Here, we define $\mathbf{\Phi}$ to be an $|S|\times n$ matrix such that $\mathbf{\Phi}[i, j] = \phi_j(s_i)$ for $s_i \in S$ and $j \in \{1, 2, \ldots, n\}$, where n is the number of features. In words, $\mathbf{\Phi}$ is a matrix representing feature vectors of the entire state space. This definition allows us to express relationships between MDPs and feature representations that are independent of an agent's history.

It will also help us to represent the state-based behavior policy transition model as a matrix. Let $\mathbf{P}_\pi$ be an $|S| \times |S|$ matrix representing the discounted state transition probabilities for a given behavior policy π: $\mathbf{P}_\pi[s, s'] = P(s'|s, \pi(s))$. Note that this matrix consists of the state–state transition model; it is distinct from the feature-based expectation models we have been using. The least-squares solution for the feature-based expected transition model is $\mathbf{F}_\pi^\mathsf{T} = (\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\mathsf{T}\mathbf{P}_\pi\mathbf{\Phi}$. This is a direct analog of Equation (5).

Using the linear algebra concept of an invariant subspace, we can express the ability of a feature set to allow for an accurate feature-based linear expected transition model of a given option.

Definition 2. *A feature space $\mathbf{\Phi}$ is* transition accurate *with respect to behavior policy π if it is subspace invariant with respect to $\mathbf{P}_\pi$, i.e., if $\mathbf{P}_\pi\mathbf{\Phi}$ is in span$(\mathbf{\Phi})$. Mathematically, $\exists\mathbf{F}_\pi$ such that*

$$\mathbf{P}_\pi\mathbf{\Phi} = \mathbf{\Phi}\mathbf{F}_\pi^\mathsf{T}. \quad (19)$$

Next, we define a similar criterion for the reward function. Let $\mathbf{r}_\pi$ be the vector of rewards corresponding to the SMDP reward function from each state while following behavior policy π: $\mathbf{r}_\pi[s] = R(s, \pi(s))$. The corresponding least-squares solution for the feature-based linear reward model is $\mathbf{b}_\pi = (\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\mathsf{T}\mathbf{r}_\pi$.

Definition 3. *A feature space $\mathbf{\Phi}$ is* reward accurate *with respect to behavior policy π if $\mathbf{r}_\pi$ is in span$(\mathbf{\Phi})$. In other words $\exists\mathbf{b}_\pi$ such that*

$$\mathbf{r}_\pi = \mathbf{\Phi}\mathbf{b}_\pi. \quad (20)$$

We will refer to Definitions 2 and 3 as the accuracy conditions for features $\mathbf{\Phi}$ with respect to behavior policy π. If these definitions hold, the feature-based linear expectation model of a behavior policy can accurately represent the true expectation of the policy's dynamics. These are strong conditions; in general, linear feature sets are used precisely because they provide a simple means of approximation. However, these conditions will allow us to motivate and define the form of composed linear models in the next section. First, we show that if transition and reward accuracy hold with respect to a given behavior policy, the above policy evaluation methods will converge to the true value function of that policy.

When using an imperfect feature representation, however, the value function of a policy cannot be exactly represented. Specifically, it will have non-zero Bellman error after convergence to the SMDP-LSTD fixed point. Theorem 2, a straightforward extension of Theorem 4.1 from Parr et al. [7],

allows us to bound the error of the converged value function as a function of the deviation from the above two accuracy conditions.

Henceforth, we drop π from the notation when it is clear from context. Define the deviation from the reward accuracy condition $\Delta_{\mathbf{r}} = \mathbf{r} - \mathbf{\Phi b}$. Similarly define the deviation from the transition accuracy condition $\Delta_{\mathbf{P}} = \mathbf{P\Phi} - \mathbf{\Phi F}^{\mathsf{T}}$. One standard measure of error in a value function is Bellman error. Define

$$\mathrm{BE}(\mathbf{\Phi}) = \mathbf{r} + \mathbf{P\Phi}\theta - \mathbf{\Phi}\theta \tag{21}$$

to be the Bellman error associated with features $\mathbf{\Phi}$ after value function parameter convergence to the SMDP-LSTD fixed point. The following theorem relates Bellman error to the deviation from the accuracy conditions.

THEOREM 2. *Given any set of features $\mathbf{\Phi}$ and behavior policy model $(\mathbf{P}, \mathbf{r})$, the value function, after convergence to $\theta = (\mathbf{I} - \mathbf{F}^{\mathsf{T}})^{-1}\mathbf{b}$, where $(\mathbf{F}, \mathbf{b})$ is a linear expectation model of the behavior policy dynamics, will have Bellman error*

$$\mathrm{BE}(\mathbf{\Phi}) = \Delta_{\mathbf{r}} + \Delta_{\mathbf{P}}\theta. \tag{22}$$

PROOF. Adapted from Parr et al. [7]:

$$\begin{aligned}
\mathrm{BE}(\Phi) &= \mathbf{r} + \mathbf{P\Phi}\theta - \mathbf{\Phi}\theta \\
&= (\Delta_{\mathbf{r}} + \mathbf{\Phi b}) + (\Delta_{\mathbf{P}} + \mathbf{\Phi F}^{\mathsf{T}})\theta - \mathbf{\Phi}\theta \\
&= \Delta_{\mathbf{r}} + \Delta_{\mathbf{P}}\theta + \mathbf{\Phi b} - \mathbf{\Phi}(\mathbf{I} - \mathbf{F}^{\mathsf{T}})\theta \\
&= \Delta_{\mathbf{r}} + \Delta_{\mathbf{P}}\theta + \mathbf{\Phi b} - \mathbf{\Phi b} \\
&= \Delta_{\mathbf{r}} + \Delta_{\mathbf{P}}\theta
\end{aligned}$$

□

It follows that for a feature set $\mathbf{\Phi}$ that is transition and reward accurate with respect to policy π, the evaluation of that policy after convergence will have zero Bellman error.

COROLLARY 1. *If $\mathbf{\Phi}$ is transition accurate w.r.t. $\mathbf{P}$ and reward accurate w.r.t. $\mathbf{r}$, the Bellman error after parameter convergence to θ will be zero.*

Theorem 2 and Corollary 1 state conditions on a feature set that lead to an accurate value function. This has strong implications for feature selection methods. However, these conditions apply only with respect to a given behavioral policy. A learning/planning agent will not know a priori what the optimal policy is, and it is ultimately the value of the optimal policy that is desired. Ideally, we would like to express conditions under which a feature set allows for good evaluations of an interesting set of policies, or more generally, all policies. In the next section, we take a step towards this goal by expressing conditions under which options may be composed to create new options for which the above conditions hold.

4.2 Compositionality

In this section, we will refer to models of options instead of models of behavior policies. Recall that the model of an option is a model of a single-option behavior policy—the policy which selects the indicated option in all states. We denote models of options using the subscript o. When an accuracy condition from Definitions 2 or 3 holds with respect to a single-option policy, we say that the accuracy condition holds with respect to the corresponding option. In order to express how option models can be composed, we first define the concept of an option composition.

DEFINITION 4. *A composition of options o_1 and o_2 is a new option o_1o_2 which follows the open-loop policy, "execute option o_1 followed by option o_2." The model of the composition $(\mathbf{P}_{o_1o_2}, \mathbf{r}_{o_1o_2})$ is equal to*

$$\mathbf{P}_{o_1o_2} = \mathbf{P}_{o_1}\mathbf{P}_{o_2} \tag{23}$$

$$\mathbf{r}_{o_1o_2} = \mathbf{r}_{o_1} + \mathbf{P}_{o_1}\mathbf{r}_{o_2}. \tag{24}$$

As shown in this definition, composing option models is a simple linear operation. A natural question to ask is whether it is possible to do a similar thing with the feature-based linear-option expectation models. We will show that if the accuracy conditions hold for the composed options, the composition of feature-based models $(\mathbf{F},\mathbf{b})$ has the same form as the composition of state-based models $(\mathbf{P},\mathbf{r})$ and that the accuracy conditions will hold for any sequence of compositions. We show the proof in three steps.

LEMMA 1. *Let $\mathbf{F}_{o_1}$ and $\mathbf{F}_{o_2}$ be the linear expectation transition models of two options and let $\mathbf{F}_{o_1o_2}$ be the linear expectation transition model of their composition. If the transition accuracy property holds for $\mathbf{\Phi}$ with respect to options o_1 and o_2, then $\mathbf{F}^{\mathsf{T}}_{o_1o_2} = \mathbf{F}^{\mathsf{T}}_{o_1}\mathbf{F}^{\mathsf{T}}_{o_2}$, and the transition accuracy property holds for $\mathbf{\Phi}$ with respect to o_1o_2.*

PROOF. By definition, $\mathbf{P}_{o_1o_2} = \mathbf{P}_{o_1}\mathbf{P}_{o_2}$. Then

$$\mathbf{P}_{o_1o_2}\mathbf{\Phi} = \mathbf{P}_{o_1}\mathbf{P}_{o_2}\mathbf{\Phi} = \mathbf{P}_{o_1}\mathbf{\Phi}\mathbf{F}^{\mathsf{T}}_{o_2} = \mathbf{\Phi}\mathbf{F}^{\mathsf{T}}_{o_1}\mathbf{F}^{\mathsf{T}}_{o_2} = \mathbf{\Phi}\mathbf{F}^{\mathsf{T}}_{o_1o_2}.$$

□

The feature-based linear reward models of two options can also be composed in a manner similar to state-based reward model composition. We use Lemma 1 in the proof.

LEMMA 2. *Let $(\mathbf{F}_{o_1}, b_{o_1})$ and $(\mathbf{F}_{o_2}, b_{o_2})$ be the linear-option expectation models of two options and let $(\mathbf{F}_{o_1o_2}, b_{o_1o_2})$ be the linear-option expectation model of their composition. If the transition and reward accuracy properties hold for $\mathbf{\Phi}$ with respect to o_1 and o_2, then $b_{o_1o_2} = b_{o_1} + \mathbf{F}^{\mathsf{T}}_{o_1}b_{o_2}$ and the reward accuracy property holds for $\mathbf{\Phi}$ with respect to the composition o_1o_2.*

PROOF. By definition, $\mathbf{r}_{o_1o_2} = \mathbf{r}_{o_1} + \mathbf{P}_{o_1}\mathbf{r}_{o_2}$. Then

$$\begin{aligned}
\mathbf{r}_{o_1o_2} &= \mathbf{r}_{o_1} + \mathbf{P}_{o_1}\mathbf{r}_{o_2} \\
&= \mathbf{\Phi}b_{o_1} + \mathbf{P}_{o_1}\mathbf{\Phi}b_{o_2} \\
&= \mathbf{\Phi}b_{o_1} + \mathbf{\Phi}\mathbf{F}^{\mathsf{T}}_{o_1}b_{o_2} \\
&= \mathbf{\Phi}(b_{o_1} + \mathbf{F}^{\mathsf{T}}_{o_1}b_{o_2}) \\
&= \mathbf{\Phi}(b_{o_1o_2}).
\end{aligned}$$

□

These two lemmas can be combined in sequence to create accurate models of arbitrarily long open-loop sequences of composed options.

THEOREM 3. *If the transition and reward accuracy properties hold for $\mathbf{\Phi}$ with respect to all options $o \in O$, then the transition and reward accuracy properties hold with respect to all sequences of compositions of $o \in O$.*

PROOF. This can be shown via a straightforward induction proof using Lemmas 1 and 2 in the inductive step. □

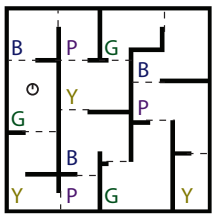

Figure 1: Continuous Rooms World

In addition to defining linear option model composition under ideal conditions, Theorem 3 has implications for the study of feature sets in general primitive-action MDPs. Because a primitive action is a special case of an option, if the accuracy conditions hold with respect to a set of primitive actions, Theorem 3 provides an algorithm for accurately evaluating arbitrary open-loop policies. Because options themselves consist of closed-loop subroutines in general, this result is broader than completely open-loop control.

5. EMPIRICAL ILLUSTRATION

Having defined linear-option expectation-models, we turn next to an empirical illustration of their use in a domain in which tabular representations of options are not feasible. We developed the Continuous Rooms World in Figure 1 as a simple extension of the discrete rooms environment from Sutton et al. [10] to a continuous state space. The world consists of a 20x20 unit building with 12 rooms separated by impenetrable walls. Additionally, the floor of each room is colored with one of 4 colors: (Y)ellow, (G)reen, (B)lue, or (P)urple. In the figure, floor colors are indicated by first letter and are separated by dashed lines.

The agent controls a small (1 unit diameter) circular wheeled robot that is capable of observing its current location $(x, y) \in [0, 20]^2$, and orientation $\psi \in [0, 360]$. In addition, it has a floor sensor capable of detecting the color of the floor beneath it. It has 3 available primitive actions: `forward`, `left`, and `right`. The `forward` action moves the agent's location 1 unit in the direction of its current orientation, offset by two-dimensional mean 0 Gaussian noise with 0.1 standard deviation. The `left` and `right` actions turn the agent 30 degrees in the specified direction. When the robot impacts a wall, its motion is stopped in the direction perpendicular to the wall. The agent is given a reward of 1 for reaching the yellow bottom-right corner room. After receiving its reward, the agent is transported to the green room on the left side of the building. At all other times, the agent receives a small negative reward of 0.01.

We tested four agents in the continuous rooms world. Each agent represents state using a feature vector ϕ of 3472 features. The first four features are binary indicator variables, one for each floor color, indicating the color of the floor beneath the center of the agent. The remaining features are radial basis features of the form $\phi_i = b \exp(-\frac{1}{2}(s - u_i)^\top \mathbf{C}(s - u_i))$, where $s = [x, y, \psi]$, $b = 20$, $\mathbf{C} = diag(\frac{1}{1.2}, \frac{1}{1.2}, \frac{1}{30})$, and the means u_i are determined by placing a radial-basis function every 1 unit in the x and y dimensions and every 30 degrees. To make the feature vector sparse, which can help computationally, any feature that would have had a value $\phi_i < 0.1$, is instead set to 0. Each agent follows an ϵ-greedy policy with $\epsilon = 0.1$. The learning rate is set to a constant $\alpha = 5 \times 10^{-4}$ for all updates. All initial value function parameters θ are initialized to the zero vector, $\mathbf{0}$. For agents that learn linear models, the reward models $\mathbf{b}$ are initialized to $\mathbf{0}$, and the expected transition models $\mathbf{F}$ are initialized to the matrix such that all entries are $1/d$, where d is the number of features.

We test 2 linear option-based agents. One is model-free and the other is model-based. The option-based agents were equipped with 4 options—one for each floor color. Each option specifies the subgoal of reaching the nearest state in which the target color is observed, terminating when the appropriate feature is set. The options are available everywhere—when already in a room of the correct color, the option's goal is to stay there. Both agents choose options ϵ-greedily, running them until termination.

The first option-based agent, called the intra-option learning agent, is model-free. It simply applies linear intra-option value learning as defined in Section 3.4 to update the behavioral Q-function after every step. The second option-based agent, the LOEM planning agent mixes learning updates with planning updates. After each time step, it makes all the relevant intra-option model/value/pseudo-value updates. Before choosing its next option, it makes one LOEM planning update for each option using that option's linear model from the current state.

We test 2 primitive-action baseline agents. One is model-free and the other model-based. Each is exactly like the corresponding option-based agent but instead only has access to primitive actions. When the intra-option value learning method above is applied to primitive actions, the equations reduce to standard Q-learning. Similarly, the intra-option model learning method reduces to the gradient update in the original linear Dyna work [12].

5.1 Evaluation Method

Options are particularly advantageous in settings in which they are *reusable*. In these settings, an agent can learn the skills in one context and apply them in another. For example, an agent can learn an option policy and transition model while solving one task and keep the option available for a later, different task in the same state space. To illustrate the ability to take advantage of such settings, we used a two-phase evaluation strategy similar to the one used by Şimşek and Barto [8]. During phase one, the *exploration phase*, agents are allowed to act and learn about the world but are not evaluated. During phase two, the *test phase*, the agents are evaluated in a control task.

For our experiment, each tested agent was given an initial exploration phase of 10^6 time steps, during which the transition dynamics were as described, but there was no task specified (no reward was given), and the agents acted randomly. During this time, each agent learned its option policies and/or transition models, if applicable. It is here that the intra-option model learning and option policy improvement methods described in Section 3.4 are used. Note that, because the agent does not have access to the reward function at this phase, the model-building agents can only learn properties of the transition function.

Next, the agent is put in a test phase of 10^5 time steps. In the test phase, the agent receives reward as normal. It is here that the intra-option value learning, reward model learning, and linear model planning algorithms are tested.

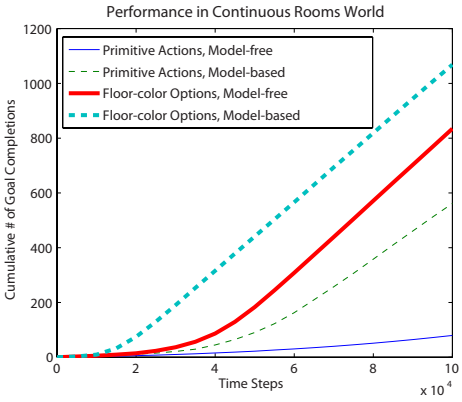

Figure 2: Performance in Continuous Rooms World

We plot the cumulative performance of each agent throughout the test phase in Figure 2. Each curve was averaged over 30 trials. Our main contribution—the linear-option model-based agent—accumulates the most reward during the test phase. The linear primitive-action model-based agent vastly out-performs the the primitive-action model-free agent, but falls short of the linear-option model-free agent. Of course, the relative performance does depend heavily on the amount of computation devoted to planning. Nevertheless, the experiment reported here does serve to illustrate that the combination of temporal and state abstraction achieved by the linear-option expectation models outperforms the solely state abstraction achieved by primitive-action linear expectation models when both are used in similar algorithms.

6. DISCUSSION

In summary, we presented the form of a linear-option expectation model. When using such models in a policy evaluation setting, we showed that TD updates lead to the same solution as does the novel SMDP-LSTD algorithm. We developed intra-option methods for learning the linear-option expectation models as well for value function learning with such models. We defined control algorithms that use either or both value function learning updates and planning updates. We extended recent work on linear feature selection to the option policy setting, showing conditions under which the value estimate of a behavioral policy will have low Bellman error after convergence. We showed that under these same conditions, when linear options are composed in an open-loop sequence to create a new option, the component option models can be easily and accurately combined. Finally, we used a continuous navigation task to illustrate the use of our linear-option expectation models and showed that they can outperform the use of primitive-action linear expectation models.

Acknowledgments

This work is supported by the Air Force Office of Scientific Research under grant FA9550-08-1-0418 as well as by NSF grant IIS 0905146. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsors

7. REFERENCES

[1] Justin A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2-3):233–246, 2002.

[2] Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[3] T. Croonenborghs, K. Driessens, and M. Bruynooghe. Learning relational options for inductive transfer in relational reinforcement learning. *Lecture Notes in Computer Science*, 4894:88, 2008.

[4] Thomas Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 1998.

[5] George Konidaris and Andy Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 895–900, 2007.

[6] George Konidaris and Andy Barto. Efficient skill learning using abstraction selection. In *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence*, pages 1107–1112, 2009.

[7] Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 752–759, New York, NY, USA, 2008. ACM.

[8] Özgür Şimşek and Andrew G. Barto. An intrinsic reward mechanism for efficient exploration. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 833–840, 2006.

[9] Vishal Soni and Satinder Singh. Using Homomorphisms to transfer options across continuous reinforcement learning domains. *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

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

[11] Richard S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In *The Seventh International Conference on Machine Learning*, pages 216–224, 1990.

[12] Richard S. Sutton, Csaba Szepesvari, Alborz Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 528–536, 2008.

[13] John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690, 1997.