Intra-Option Learning about Temporally Abstract Actions

Richard S. Sutton
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
rich@cs.umass.edu

Doina Precup
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
dprecup@cs.umass.edu

Satinder Singh
Department of Computer Science
University of Colorado
Boulder, CO 80309-0430
baveja@cs.colorado.edu

Abstract

Several researchers have proposed modeling temporally abstract actions in reinforcement learning by the combination of a policy and a termination condition, which we refer to as an *option*. Value functions over options and models of options can be learned using methods designed for semi-Markov decision processes (SMDPs). However, all these methods require an option to be executed to termination. In this paper we explore methods that learn about an option from small fragments of experience consistent with that option, even if the option itself is not executed. We call these methods *intra-option* learning methods because they learn from experience *within* an option. Intra-option methods are sometimes much more efficient than SMDP methods because they can use off-policy temporal-difference mechanisms to learn simultaneously about all the options consistent with an experience, not just the few that were actually executed. In this paper we present intra-option learning methods for learning value functions over options and for learning multi-time models of the consequences of options. We present computational examples in which these new methods learn much faster than SMDP methods and learn effectively when SMDP methods cannot learn at all. We also sketch a convergence proof for intra-option value learning.

1 Introduction

Learning, planning, and representing knowledge at multiple levels of temporal abstraction remain key challenges for AI. Recently, several researchers have begun to address these challenges within the framework of reinforcement learning and Markov decision processes (MDPs) (e.g., Singh, 1992a,b; Kaelbling, 1993; Lin, 1993; Dayan & Hinton, 1993; Thrun and Schwartz, 1995; Sutton, 1995; Huber and Grupen, 1997; Kalmár, Szepesvári, and Lörincz, 1997; Dietterich, 1998; Parr and Russell, 1998; Precup, Sutton, and Singh 1997, 1998a,b). This framework is appealing because of its general goal formulation, applicability to stochastic environments, and ability to use sample or simulation models (e.g., see Sutton and Barto, 1998). Extensions of MDPs to *semi-Markov decision processes* (SMDPs) provide a way to model temporally abstract actions, as we summarize in Sections 3 and 4 below. Common to much of this recent work is the modeling of a temporally extended action as a policy (controller) and a condition for terminating, which we together refer to as an *option*. Options are a flexible way of representing temporally extended courses of action such that they can be used interachangeably with primitive actions in existing learning and planning methods (Sutton, Precup, and Singh, in preparation).

In this paper we explore ways for learning about options using a class of off-policy, temporal-difference methods that we call *intra-option* learning methods. Intra-option methods look inside options to learn about them even when only a single action is taken that is consistent with them. Whereas SMDP methods treat options as indivisible black boxes, intra-option methods attempt to take advantage of their internal structure to speed learning. Intra-option methods were introduced by Sutton (1995), but only for a pure prediction case, with a single policy.

The structure of this paper is as follows. First we introduce the basic notation of reinforcement learning, options and models of options. In Section 4 we briefly review SMDP methods for learning value functions over options and thus how to select among options. Our new results are in Sections 5–7. Section 5 introduces an intra-option method for learning value functions and sketches a proof of its convergence. Computational experiments comparing it with SMDP methods are presented in Section 6. Section 7 concerns methods for learning *models* of options, as are used in planning: we introduce an intra-option method and illustrate its advantages in computational experiments.

2 Reinforcement Learning (MDP) Framework

In the reinforcement learning framework, a learning *agent* interacts with an *environment* at some discrete, lowest-level time scale $t = 0, 1, 2, \ldots$. At each time step, the agent perceives the state of the environment, $s_t \in \mathcal{S}$, and on that basis chooses a primitive action, $a_t \in \mathcal{A}_{s_t}$. In response to a_t, the environment produces one step later a numerical reward, $r_{t+1} \in \Re$, and a next state, s_{t+1}. We denote the union of the action sets by $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$. If $\mathcal{S}$ and $\mathcal{A}$, are finite, then the environment's transition dynamics are modeled by one-step state-transition probabilities, and one-step expected rewards,

$$\begin{aligned} p^a_{ss'} &= \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \quad \text{and} \\ r^a_s &= E\{r_{t+1} \mid s_t = s, a_t = a\}, \end{aligned}$$

for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ (it is understood here that $p^a_{ss'} = 0$ for $a \notin \mathcal{A}_s$). These two sets of quantities together constitute the *one-step model* of the environment.

The agent's objective is to learn a policy π, which is a mapping from states to probabilities of taking each action, that maximizes the expected discounted future reward from each state s:

$$V^\pi(s) = E\Big\{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \Bigm| s_t = s, \pi\Big\},$$

where $\gamma \in [0, 1)$ is a *discount-rate* parameter. The quantity $V^\pi(s)$ is called the *value* of state s under policy π, and V^π is called the value function for policy π. The optimal value of a state is denoted

$$V^*(s) = \max_\pi V^\pi(s).$$

Particularly important for learning methods is a parallel set of value functions for state–action pairs rather than for states. The value of taking action a in state s under policy π, denoted $Q^\pi(s, a)$, is the expected discounted future reward starting in s, taking a, and henceforth following π:

$$Q^\pi(s, a) = E\Big\{r_{t+1} + \gamma r_{t+1} + \cdots \Bigm| s_t = s, a_t = a, \pi\Big\}.$$

This is known as the *action-value function* for policy π. The *optimal* action-value function is

$$Q^*(s, a) = \max_\pi Q^\pi(s, a).$$

The action value functions satisfy the Bellman equations:

$$Q^\pi(s, a) = r^a_s + \gamma \sum_{s'} p^a_{ss'} \sum_{a'} \pi(s', a') Q^\pi(s', a') \quad (1)$$

$$Q^*(s, a) = r^a_s + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q^*(s', a'). \quad (2)$$

3 Options

We use the term *options* for our generalization of primitive actions to include temporally extended courses of action. In this paper, we focus on *Markov options*, which consist of three components: a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, a termination condition $\beta : \mathcal{S} \mapsto [0, 1]$, and an input set $\mathcal{I} \subseteq \mathcal{S}$. An option $\langle \mathcal{I}, \pi, \beta \rangle$ is available in state s if and only if $s \in \mathcal{I}$. If the option is taken, then actions are selected according to π until the option terminates stochastically according to β. In particular, if the option taken in state s_t is Markov, then the next action a_t is selected according to the probability distribution $\pi(s_t, \cdot)$. The environment then makes a transition to state s_{t+1}, where the option either terminates, with probability $\beta(s_{t+1})$, or else continues, determining a_{t+1} according to $\pi(s_{t+1}, \cdot)$, possibly terminating in s_{t+2} according to $\beta(s_{t+2})$, and so on. When the option terminates, then the agent has the opportunity to select another option.

The input set and termination condition of an option together restrict its range of application in a potentially useful way. In particular, they limit the range over which the option's policy needs to be defined. For example, a handcrafted policy π for a mobile robot to dock with its battery charger might be defined only for states $\mathcal{I}$ in which the battery charger is within sight. The termination condition β would be defined to be 1 outside of $\mathcal{I}$ and when the robot is successfully docked. For Markov options it is natural to assume that all states where an option might continue are also states where the option might be taken (i.e., that $\{s : \beta(s) < 1\} \subseteq \mathcal{I}$). In this case, π needs to be defined only over $\mathcal{I}$ rather than over all of $\mathcal{S}$.

Given a set of options, their input sets implicitly define a set of available options $\mathcal{O}_s$ for each state $s \in \mathcal{S}$. The sets $\mathcal{O}_s$ are much like the sets of available actions, $\mathcal{A}_s$. We can unify these two kinds of sets by noting that actions can be considered a special case of options. Each action a corresponds to an option that is available whenever a is available ($\mathcal{I} = \{s : a \in \mathcal{A}_s\}$), that always lasts exactly one step ($\beta(s) = 1, \ \forall s \in \mathcal{S}$), and that selects a everywhere ($\pi(s, a) = 1, \ \forall s \in \mathcal{I}$). Thus, we can consider the agent's choice at each time to be entirely among options, some of which persist for a single time step, others which are more temporally extended. We refer to the former as *one-step* or *primitive* options and the latter as *multi-step* options.

We now consider Markov policies over options, $\mu : \mathcal{S} \times \mathcal{O} \mapsto [0,1]$, and their value functions. When initiated in a state s_t, such a policy μ selects an option $o \in \mathcal{O}_{s_t}$ according to probability distribution $\mu(s_t, \cdot)$. The option o is taken in s_t, determining actions until it terminates in s_{t+k}, at which point a new option is selected, according to $\mu(s_{t+k}, \cdot)$, and so on. In this way a policy over options, μ, determines a policy over actions, or *flat policy*, $\pi = f(\mu)$. Henceforth we use the unqualified term *policy* for Markov policies over options, which include Markov flat policies as a special case.

Note, however, that $f(\mu)$ is typically not Markov because the action taken in a state depends on which option is being taken at the time, not just on the state. We define the value of a state s under a general flat policy π as the expected return if the policy is started in s:

$$V^\pi(s) \overset{\text{def}}{=} E\Big\{r_{t+1} + \gamma r_{t+2} + \cdots \;\Big|\; \mathcal{E}(\pi, s, t)\Big\},$$

where $\mathcal{E}(\pi, s, t)$ denotes the event of π being initiated in s at time t. The value of a state under a general policy (i.e., a policy over options) μ can then be defined as the value of the state under the corresponding flat policy: $V^\mu(s) \overset{\text{def}}{=} V^{f(\mu)}(s)$.

It is natural to also generalize the action-value function to an *option*-value function. We define $Q^\mu(s,o)$, the value of taking option o in state $s \in \mathcal{I}$ under policy μ, as

$$Q^\mu(s,o) \overset{\text{def}}{=} E\Big\{r_{t+1} + \gamma r_{t+2} + \cdots \;\Big|\; \mathcal{E}(o\mu, s, t)\Big\},$$

where $o\mu$, the *composition* of o and μ, denotes the policy that first follows o until it terminates and then initiates μ in the resultant state.

Options are closely related to the actions in a special kind of decision problem known as a *semi-Markov decision process*, or *SMDP* (e.g., see Puterman, 1994). Any fixed set of options for a given MDP defines a new SMDP overlaid on the MDP. The appropriate form of model for options, analogous to the r_s^a and $p_{ss'}^a$ defined earlier for actions, is known from existing SMDP theory. For each state in which an option may be started, this kind of model predicts the state in which the option will terminate and the total reward received along the way. These quantities are discounted in a particular way. For any option o, let $\mathcal{E}(o, s, t)$ denote the event of o being taken in state s at time t. Then the reward part of the model of o for state s is

$$r_s^o = E\Big\{r_{t+1} + \gamma r_{t+2} ... + \gamma^{k-1} r_{t+k} \;\Big|\; \mathcal{E}(o, s, t)\Big\}, \quad (3)$$

where $t + k$ is the random time at which o terminates. The state-prediction part of the model of o for state s is

$$\begin{aligned} p_{ss'}^o &= \sum_{j=0}^{\infty} \gamma^j \Pr\{k = j, s_{t+j} = s' \mid \mathcal{E}(o, s, t)\} \\ &= E\{\gamma^k \delta_{s' s_{t+k}} \mid \mathcal{E}(o, s, t)\}, \end{aligned} \quad (4)$$

for all $s' \in \mathcal{S}$, under the same conditions, where $\delta_{ss'}$ is an identity indicator, equal to 1 if $s = s'$, and equal to 0 else. Thus, $p_{ss'}^o$ is a combination of the likelihood that s' is the state in which o terminates together with a measure of how delayed that outcome is relative to γ. We call this kind of model a *multi-time model* because it describes the outcome of an option not at a single time but at potentially many different times, appropriately combined.

4 SMDP Learning Methods

Using multi-time models of options we can write Bellman equations for general policies and options. For example, the Bellman equation for the value of option o in state $s \in \mathcal{I}$ under a Markov policy μ is

$$Q^\mu(s,o) = r_s^o + \sum_{s'} p_{ss'}^o \sum_{o' \in \mathcal{O}_s} \mu(s', o') Q^\mu(s', o'). \quad (5)$$

The *optimal* value functions and *optimal* Bellman equations can also be generalized to options and to policies over options. Of course, the conventional optimal value functions V^* and Q^* are not affected by the introduction of options; one can ultimately do just as well with primitive actions as one can with options. Nevertheless, it is interesting to know how well one can do with a restricted set of options that does not include all the actions. For example, one might first consider only high-level options in order to find an approximate solution quickly. Let us denote the restricted set of options by $\mathcal{O}$ and the set of all policies that select only from $\mathcal{O}$ by $\Pi(\mathcal{O})$. Then the optimal value function given that we can select only from $\mathcal{O}$ is

$$\begin{aligned} V_{\mathcal{O}}^*(s) &\overset{\text{def}}{=} \max_{\mu \in \Pi(\mathcal{O})} V^\mu(s) && (6) \\ &= \max_{o \in \mathcal{O}} E\Big\{r + \gamma^k V_{\mathcal{O}}^*(s') \;\Big|\; \mathcal{E}(o, s)\Big\} && (7) \end{aligned}$$

where $\mathcal{E}(o, s)$ denotes the event of starting the execution of option o in state s, k is the random numbner opf steps elapsing during o, s' is the resulting next state, and r is the cumulative discounted reward received along the way. The optimal option values are defined as:

$$\begin{aligned} Q_{\mathcal{O}}^*(s,o) &\overset{\text{def}}{=} \max_{\mu \in \Pi(\mathcal{O})} Q^\mu(s,o) && (8) \\ &= E\Big\{r + \gamma^k \max_{o' \in \mathcal{O}} Q_{\mathcal{O}}^*(s', o') \;\Big|\; \mathcal{E}(o, s)\Big\} && (9) \end{aligned}$$

Given a set of options, $\mathcal{O}$, a corresponding *optimal policy*, denoted $\mu^*_{\mathcal{O}}$, is any policy that achieves $V^*_{\mathcal{O}}$, i.e., for which $V^{\mu^*_{\mathcal{O}}}(s) = V^*_{\mathcal{O}}(s)$ for all states $s \in \mathcal{S}$. If $V^*_{\mathcal{O}}$ and models of the options are known, then optimal policies can be formed by choosing in any proportion among the maximizing options in (7). Or, if $Q^*_{\mathcal{O}}$ is known, then optimal policies can be formed by choosing in each state s in any proportion among the options o for which $Q^*_{\mathcal{O}}(s,o) = \max_{o'} Q^*_{\mathcal{O}}(s,o')$. Thus, computing approximations to $V^*_{\mathcal{O}}$ or $Q^*_{\mathcal{O}}$ become the primary goals of planning and learning methods with options.

The problem of finding the optimal value functions for a set of options can be addressed by learning methods. Because an MDP augmented by options forms an SMDP, we can apply SMDP learning methods as developed by Bradtke and Duff (1995), Parr and Russell (1998), Parr (in preparation), Mahadevan et al. (1997), and McGovern, Sutton and Fagg (1997). In these methods, each option is viewed as an indivisible, opaque unit. After the execution of option o is started in state s, we next jump to the state s' in which it terminates. Based on this experience, an estimate $Q(s,o)$ of the optimal option-value function is updated. For example, the SMDP version of one-step Q-learning (Bradtke and Duff, 1995), which we call *one-step SMDP Q-learning*, updates after each option termination by

$$Q(s,o) \leftarrow Q(s,o) + \alpha \left[r + \gamma^k \max_{o' \in \mathcal{O}} Q(s',o') - Q(s,o) \right],$$

where k is the number of time steps elapsing between s and s', r is the cumulative discounted reward over this time, and it is implicit that the step-size parameter α may depend arbitrarily on the states, option, and time steps. The estimate $Q(s,o)$ converges to $Q^*_{\mathcal{O}}(s,o)$ for all $s \in \mathcal{S}$ and $o \in \mathcal{O}$ under conditions similar to those for conventional Q-learning (Parr, in preparation).

5 Intra-Option Value Learning

One drawback to SMDP learning methods is that they need to execute an option to termination before they can learn about it. Because of this, they can only be applied to one option at a time—the option that is executing at that time. More interesting and potentially more powerful methods are possible by taking advantage of the structure inside each option. In particular, if the options are Markov and we are willing to look *inside* them, then we can use special temporal-difference methods to learn usefully about an option before the option terminates. This is the main idea behind *intra-option* methods.

Intra-option methods are examples of *off-policy* learning methods (Sutton and Barto, 1998) in that they learn about the consequences of one policy while actually behaving according to another, potentially different policy. Intra-option methods can be used to learn simultaneously about many different options from the same experience. Moreover, they can learn about the values of executing options *without ever executing* those options.

Intra-option methods for value learning are potentially more efficient than SMDP methods because they extract more training examples from the same experience. For example, suppose we are learning to approximate $Q^*_{\mathcal{O}}(s,o)$ and that o is Markov. Based on an execution of o from t to $t+k$, SMDP methods extract a single training example for $Q^*_{\mathcal{O}}(s,o)$. But because o is Markov, it is, in a sense, also initiated at each of the steps between t and $t+k$. The jumps from each intermediate s_{t+i} to s_{t+k} are also valid experiences with o, experiences that can be used to improve estimates of $Q^*_{\mathcal{O}}(s_{t+i},o)$. Or consider an option that is very similar to o and which would have selected the same actions, but which would have terminated one step later, at $t+k+1$ rather than at $t+k$. Formally this is a different option, and formally it *was not executed*, yet all this experience could be used for learning relevant to it. In fact, an option can often learn something from experience that is only slightly related (occasionally selecting the same actions) to what would be generated by executing the option. This is the idea of off-policy training—to make full use of whatever experience occurs in order to learn as much possible about all options, irrespective of their role in generating the experience. To make the best use of experience we would like an off-policy and intra-option version of Q-learning.

It is convenient to introduce new notation for the value of a state–option pair given that the option is Markov and executing upon *arrival* in the state:

$$U^*_{\mathcal{O}}(s,o) = (1-\beta(s))Q^*_{\mathcal{O}}(s,o) + \beta(s) \max_{o' \in \mathcal{O}} Q^*_{\mathcal{O}}(s,o'),$$

Then we can write Bellman-like equations that relate $Q^*_{\mathcal{O}}(s,o)$ to expected values of $\tilde{Q}^*_{\mathcal{O}}(s',o)$, where s' is the immediate successor to s after initiating Markov option $o = \langle \mathcal{I}, \pi, \beta \rangle$ in s:

$$\begin{aligned} Q^*_{\mathcal{O}}(s,o) &= \sum_{a \in \mathcal{A}_s} \pi(s,a) E\left\{ r + \gamma U^*_{\mathcal{O}}(s',o) \,\Big|\, s,a \right\} \\ &= \sum_{a \in \mathcal{A}_s} \pi(s,a) \left[r^a_s + \sum_{s'} p^a_{ss'} U^*_{\mathcal{O}}(s',o) \right], \end{aligned}$$

where r is the immediate reward upon arrival in s'. Now consider learning methods based on this Bellman equation. Suppose action a_t is taken in state s_t to produce next state s_{t+1} and reward r_{t+1}, and that a_t was selected in a way consistent with the Markov policy π of an option

$o = \langle \mathcal{I}, \pi, \beta \rangle$. That is, suppose that a_t was selected according to the distribution $\pi(s_t, \cdot)$. Then the Bellman equation above suggests applying the off-policy one-step temporal-difference update:

$$Q(s_t, o) \leftarrow Q(s_t, o) + \alpha \Big[(r_{t+1} + \gamma U(s_{t+1}, o)) - Q(s_t, o) \Big],$$

where

$$U(s, o) = (1 - \beta(s)) Q(s, o) + \beta(s) \max_{o' \in \mathcal{O}} Q(s, o')$$

The method we call *one-step intra-option Q-learning* applies this update rule to every option o consistent with every action taken a_t.

Theorem 1 (Convergence of intra-option Q-learning) *For any set of deterministic Markov options $\mathcal{O}$, one-step intra-option Q-learning converges w.p.1 to the optimal Q-values, $Q^*_{\mathcal{O}}$, for every option, regardless of what options are executed during learning, provided every primitive action gets executed in every state infinitely often.*

Proof: (Sketch) On experiencing $\langle s, a, r, s' \rangle$, for every option o that picks action a in state s, intra-option Q-learning performs the following update:

$$Q(s, o) \leftarrow Q(s, o) + \alpha(s, o)[r + \gamma U(s', o) - Q(s, o)].$$

Let a be the action selection by deterministic Markov option $o = \langle \mathcal{I}, \pi, \beta \rangle$. Our result follows directly from Theorem 1 of Jaakkola et al. (1994) and the observation that the expected value of the update operator $r + \gamma U(s', o)$ yields a contraction, as shown below:

$$\begin{aligned}
&|E\{r + \gamma U(s', o)\} - Q^*_{\mathcal{O}}(s, o)| \\
&= |r^a_s + \sum_{s'} p^a_{ss'} U(s', o) - Q^*_{\mathcal{O}}(s, o)| \\
&= |r^a_s + \sum_{s'} p^a_{ss'} U(s', o) - r^a_s - \sum_{s'} p^a_{ss'} U^*_{\mathcal{O}}(s', o)| \\
&\leq \ |\sum_{s'} p^a_{ss'} \Big[(1 - \beta(s'))(Q(s', o) - Q^*_{\mathcal{O}}(s', o)) \\
&\quad + \beta(s')(\max_{o' \in \mathcal{O}} Q(s', o') - \max_{o' \in \mathcal{O}} Q^*_{\mathcal{O}}(s', o')) \Big] | \\
&\leq \ \sum_{s'} p^a_{ss'} \max_{s'', o''} |Q(s'', o'') - Q^*_{\mathcal{O}}(s'', o'')| \\
&\leq \ \gamma \max_{s'', o''} |Q(s'', o'') - Q^*_{\mathcal{O}}(s'', o'')| \qquad \diamond
\end{aligned}$$

6 Illustrations of Intra-Option Value Learning

As an illustration of intra-option value-learning, we used the gridworld environment shown in Figure 1. The cells of the grid correspond to the states of the environment. From any state the agent can perform one of four actions, `up`, `down`, `left` or `right`, which have a stochastic effect. With probability 2/3, the actions cause the agent to move one cell in the corresponding direction, and with probability 1/3, the agent moves instead in one of the other three directions, each with 1/9 probability. If the movement would take the agent into a wall, then the agent remains in the same cell. There are small negative rewards for each action, with means uniformly distributed between 0 and -1. The rewards are also perturbed by gaussian noise with standard deviation 0.1. The environment also has a goal state, labeled “G”. A complete trip from a random start state to the goal state is called an *episode*. When the agent enters “G”, it gets a reward of 1 and the episode ends. In all the experiments the discount parameter was $\gamma = 0.9$ and all the initial value estimates were 0.

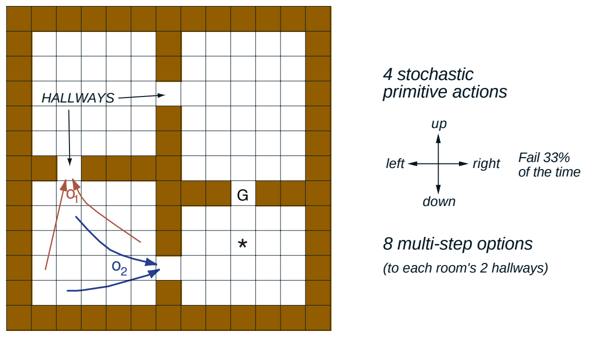


Figure 1: The rooms example is a gridworld environment with stochastic cell-to-cell actions and room-to-room hallway options. Two of the hallway options are suggested by the arrows labeled o_1 and o_2. The label G indicates the location used as a goal.

In each of the four rooms we provide two built-in *hallway options* designed to take the agent from anywhere within the room to one of the two hallway cells leading out of the room. The policies underlying the options follow the shortest expected path to the hallway.

For the first experiment, we applied the intra-option method in this environment without selecting the hallway options. In each episode, the agent started at a random state in the environment and thereafter selected primitive actions randomly, with equal probability. On every transition, the update (5) was applied first to the primitive action taken, then to any of the hallway options that were consistent with it. The hallway options were updated in clockwise order, starting from any hallways that faced up from the current state. The value of the step-size parameter was $\alpha = 0.01$.

This is a case in which SMDP methods would not be able to

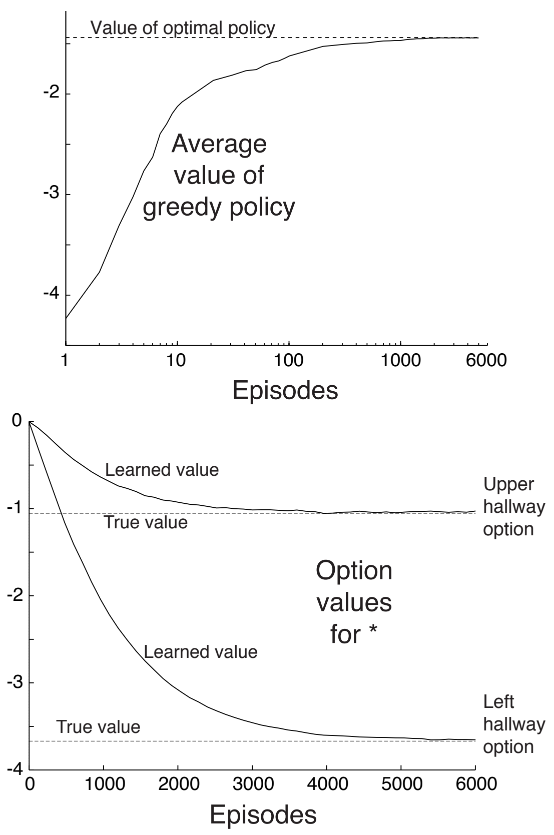


Figure 2: The learning of option values by intra-option methods without ever selecting the options. The value of the greedy policy goes to the optimal value (upper panel) as the learned values approach the correct values (as shown for one state, in the lower panel).

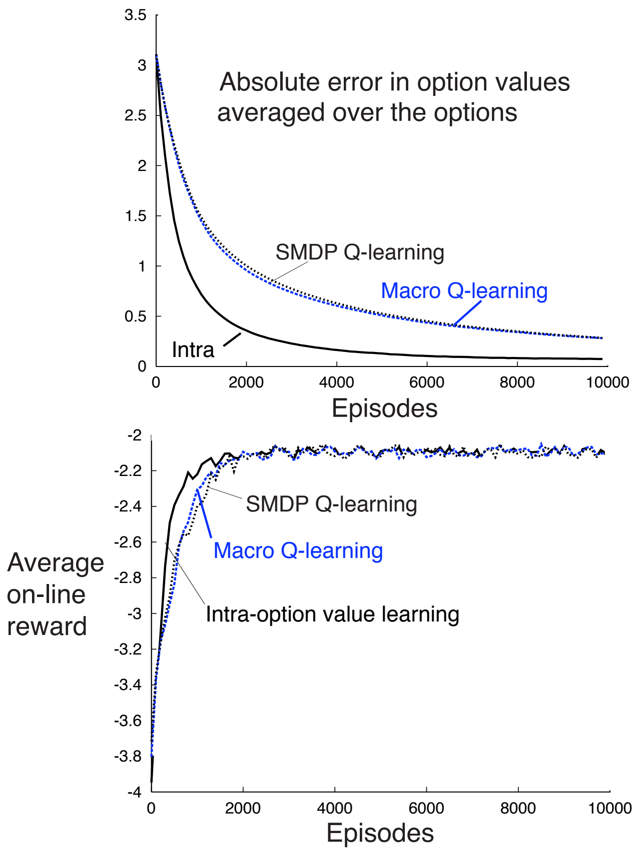


Figure 3: Comparison of SMDP, intra-option and macro Q-learning. Intra-option methods converge faster to the correct values.

learn anything about the hallway options, because these options are never executed. However, the intra-option method learned the values of these actions effectively, as shown in Figure 2. The upper panel shows the value of the greedy policy learned by the intra-option method, averaged over $\mathcal{I}$ and over 30 repetitions of the whole experiment. The lower panel shows the correct and learned values for the two hallway options that apply in the state marked $*$ in Figure 1. Similar convergence to the true values was observed for all the other states and options.

So far we have illustrated the effectiveness of intra-option learning in a context in which SMDP methods do not apply. How do intra-option methods compare to SMDP methods when both are applicable? In order to investigate this question, we used the same environment, but now we allowed the agent to choose among the hallway options as well as the primitive actions, which were treated as one-step options. In this case, SMDP methods can be applied, since all the options are actually executed. We experimented with two SMDP methods: one-step SMDP Q-learning (Bradtke and Duff, 1995) and a hierarchical form of Q-learning called *macro Q-learning* (McGovern, Sutton and Fagg, 1997). The difference between the two methods is that, when taking a multi-step option, SMDP Q-learning only updates the value of that option, whereas macro Q-learning also updates the values of the one-step options (actions) that were taken along the way.

In this experiment, options were selected not at random, but in an ϵ-greedy way dependent on the current option-value estimates. That is, given the current estimates $Q(s,o)$, let $o^* = \arg\max_{o \in \mathcal{O}_s} Q(s,o)$ denote the best valued action (with ties broken randomly). Then the policy used to select options was

$$\mu(s,o) = \begin{cases} 1-\epsilon+\frac{\epsilon}{|\mathcal{O}_s|} & \text{if } o = o^* \\ \frac{\epsilon}{|\mathcal{O}_s|} & \text{otherwise,} \end{cases}$$

for all $s \in \mathcal{S}$ and $o \in \mathcal{O}$. The probability of a random action, ϵ, was set at 0.1 in all cases. For each algorithm, we tried step-size values of $\alpha = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, and $\frac{1}{16}$ and then picked the best one.

Figure 3 shows two measures of the performance of the learning algorithms. The upper panel shows the average absolute error in the estimates of $Q^*_{\mathcal{O}}$ for the hallway options, averaged over the input sets $\mathcal{I}$, the eight hallway options, and 30 repetitions of the whole experiment. The intra-option method showed significantly faster learning than any of the SMDP methods. The lower panel shows the quality of the policy executed by each method, measured as the average reward over the state space. The intra-option method was also the fastest to learn by this measure.

7 Intra-Option Model Learning

In this section, we consider intra-option methods for learning multi-time models of options, r^o_s and $p^o_{ss'}$, given knowledge of the option (i.e., of its π, β, and $\mathcal{I}$). Such models are used in planning methods (e.g., Precup, Sutton, and Singh, 1997, 1998a,b).

The most straightforward approach to learning the model of an option is to execute the option to termination many times in each state s, recording the resultant next states s', cumulative discounted rewards r, and elapsed times k. These outcomes can then be averaged to approximate the expected values for r^o_s and $p^o_{ss'}$ given by (3) and (4). For example, an incremental learning rule for this could update its estimates $\hat{r}^o_s$ and $\hat{p}^o_{sx}$, for all $x \in \mathcal{S}$, after each execution of o in state s, by

$$\hat{r}^o_s = \hat{r}^o_s + \alpha[r - \hat{r}^o_s], \quad \text{and} \tag{10}$$
$$\hat{p}^o_{sx} = \hat{p}^o_{sx} + \alpha[\gamma^k \delta_{xs'} - \hat{p}^o_{sx}], \tag{11}$$

where the step-size parameter, α, may be constant or may depend on the state, option, and time. For example, if α is 1 divided by the number of times that o has been experienced in s, then these updates maintain the estimates as sample averages of the experienced outcomes. However the averaging is done, we call these *SMDP model-learning methods* because, like SMDP value-learning methods, they are based on jumping from initiation to termination of each option, ignoring what might happen along the way. In the special case in which o is a primitive action, note that SMDP model-learning methods reduce exactly to those used to learn conventional one-step models of actions.

Now let us consider intra-option methods for model learning. The idea is to use Bellman equations for the model, just as we used the Bellman equations in the case of learning value functions. The correct model of a Markov option $o = \langle \mathcal{I}, \pi, \beta \rangle$ is related to itself by

$$r^o_s = \sum_{a \in \mathcal{A}_s} \pi(s,a) E\Big\{ r + \gamma(1 - \beta(s')) r^o_{s'} \Big\} \tag{12}$$
$$= \sum_{a \in \mathcal{A}_s} \pi(s,a) \left[r^a_s + \sum_{s'} p^a_{ss'} (1 - \beta(s')) r^o_{s'} \right] \tag{13}$$

where r and s' are the reward and next state given that action a is taken in state s, and

$$p^o_{sx} = \sum_{a \in \mathcal{A}_s} \pi(s,a) \gamma E\Big\{ (1 - \beta(s')) p^o_{s'x} + \beta(s') \delta_{s'x} \Big\}$$
$$= \sum_{a \in \mathcal{A}_s} \pi(s,a) \sum_{s'} p^a_{ss'} (1 - \beta(s')) p^o_{s'x} + \beta(s') \delta_{s'x}$$

for all $s, x \in \mathcal{S}$. How can we turn these Bellman equations into update rules for learning the model? First consider that action a_t is taken in s_t and that the way it was selected is consistent with $o = \langle \mathcal{I}, \pi, \beta \rangle$, that is, that a_t was selected with the distribution $\pi(s_t, \cdot)$. Then the Bellman equations above suggest the temporal-difference update rules

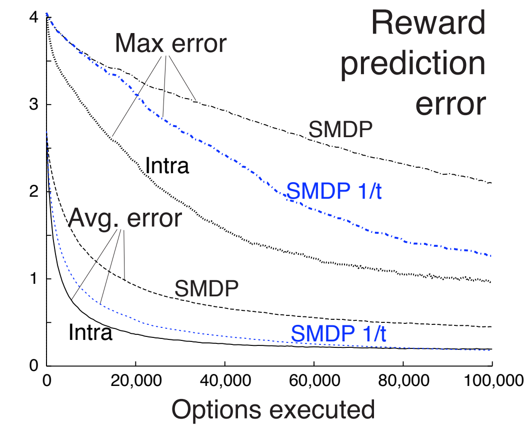
$$\hat{r}^o_{s_t} \leftarrow \hat{r}^o_{s_t} + \alpha \left[r_{t+1} + \gamma(1 - \beta(s_{t+1})) \hat{r}^o_{s_{t+1}} - \hat{r}^o_{s_t} \right] \tag{14}$$

and

$$\hat{p}^o_{s_t x} \leftarrow \hat{p}^o_{s_t x} + \alpha[\gamma(1 - \beta(s_{t+1})) \hat{p}^o_{s_{t+1} x} + \gamma \beta(s_{t+1}) \delta_{s_{t+1} x} - \hat{p}^o_{s_t x}], \tag{15}$$

where $\hat{p}^o_{ss'}$ and $\hat{r}^o_s$ are the estimates of $p^o_{ss'}$ and r^o_s, respectively, and α is a positive step-size parameter. The method we call *one-step intra-option model learning* applies these updates to every option consistent with every action taken. Of course, this is just the simplest intra-option model-learning method. Others may be possible using eligibility traces and standard tricks for off-policy learning (see Sutton, 1995; Sutton and Barto, 1998).

Intra-option methods for model learning have advantages over SMDP methods similar to those we saw earlier for value-learning methods. As an illustration, consider the application of SMDP and intra-option model-learning methods to the rooms example. We assume that the eight hallway options are given as before, but now we assume that their models are not given and must be learned. Experience is generated by selecting randomly in each state among the two possible options and four possible actions, with no goal state. In the SMDP model-learning method, equations (10) and (11) were applied whenever an option was selected, whereas, in the intra-option model-learning method, equations (14) and (15) were applied on every step to all options that were consistent with the action taken on that step. In this example, all options are deterministic, so consistency



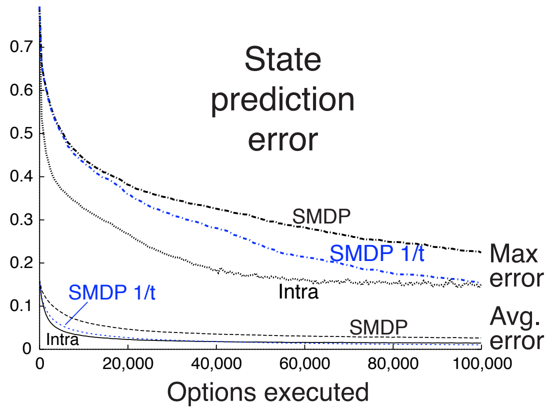


Figure 4: Learning curves for model learning by SMDP and intra-option methods.

with the action selected means simply that the option would have selected that action.

For the SMDP method, the step-size parameter was varied so that the model estimates were sample averages, which should give fastest learning. The results of this method are labeled "SMDP 1/t" on the graphs. We also looked at results using a fixed learning rate. In this case and for the intra-option method we tried step-size values of $\alpha = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, and $\frac{1}{16}$, and picked the best value for each method. Figure 4 shows the learning curves for all three methods, using the best α values, when a fixed alpha was used. The upper panel shows the average and maximum absolute error in the reward predictions, and the lower panel shows the average absolute error and the maximum absolute error in the transition predictions, averaged over the eight options and over 30 independent runs. The intra-option method approached the correct values more rapidly than the SMDP methods.

8 Closing

The theoretical and empirical results presented in this paper suggest that intra-option methods provide an efficient way for taking advantage of the structure inside an option. Intra-option methods use experience with a single action to update the value or model for all the options that are consistent with that action. In this way they make much more efficient use of the experience than SMDP methods, which treat options as indivisible units. In the future, we plan to extend these algorithms for the case of non-Markov options, and to combine them with eligibility traces.

Acknowledgements

The authors acknowledge the help of their the colleagues Amy McGovern, Andy Barto, Csaba Szepesvári, András Lörincz, Ron Parr, Tom Dietterich, Andrew Fagg, Leo Zelevinsky and Manfred Huber. We also thank Andy Barto, Paul Cohen, Robbie Moll, Mance Harmon, Sascha Engelbrecht, and Ted Perkins for helpful reactions and constructive criticism. This work was supported by NSF grant ECS-9511805 and grant AFOSR-F49620-96-1-0254, both to Andrew Barto and Richard Sutton. Doina Precup also acknowledges the support of the Fulbright foundation. Satinder Singh was supported by NSF grant IIS-9711753.

References

Bradtke, S. J. & Duff, M. O. (1995). Reinforcement learning methods for continuous-time Markov decision problems. In *Advances in Neural Information Processing Systems 7* (pp. 393–400). MIT Press.

Dayan, P. & Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5* (pp. 271–278). Morgan Kaufmann.

Dietterich, T. G. (1998). The MAXQ method for hierarchical reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann.

Huber, M. & Grupen, R. A. (1997). A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems*, *22(3-4)*, 303–315.

Jaakkola, T., Jordan, M. & Singh, S. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, *6(6)*, 1185–1201.

Kaelbling, L. P. (1993). Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the*

Tenth International Conference on Machine Learning (pp. 167–173). Morgan Kaufmann.

Kalmár, Z., Szepesvári, C. & Lörincz, A. (1997). Module based reinforcement learning for a real robot. In *Proceedings of the Sixth European Workshop on Learning Robots* (pp. 22–32).

Lin, L.-J. (1993). *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University.

Mahadevan, S., Marchallek, N., Das, T. K. & Gosavi, A. (1997). Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 202–210). Morgan Kaufmann.

McGovern, A., Sutton, R. S. & Fagg, A. H. (1997). Roles of macro-actions in accelerating reinforcement learning. In *Grace Hopper Celebration of Women in Computing* (pp. 13–17).

Parr, R. (in preparation). *Hierarchical Control and learning for Markov decision processes*. PhD thesis, Berkeley University. Chapter 3.

Parr, R. & Russel, S. (1998). Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10*. MIT Press.

Precup, D., Sutton, R. S. & Singh, S. (1997). Planning with closed-loop macro actions. In *Working Notes of the AAAI Fall Symposium '97 on Model-directed Autonomous Systems* (pp. 70–76).

Precup, D., Sutton, R. S. & Singh, S. (1998a). Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems 10*. MIT Press.

Precup, D., Sutton, R. S. & Singh, S. (1998b). Theoretical results on reinforcement learning with temporally abstract options. In *Machine Learning: ECML98. 10th European Conference on Machine Learning, Chemnitz, Germany, April 1998. Proceedings* (pp. 382–393). Springer Verlag.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Singh, S. P. (1992a). Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 202–207). MIT/AAAI Press.

Singh, S. P. (1992b). Scaling reinforcement learning by learning variable temporal resolution models. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 406–415). Morgan Kaufmann.

Sutton, R. S. (1995). TD models: Modeling the world as a mixture of time scales. In *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 531–539). Morgan Kaufmann.

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

Sutton, R. S., Precup, D. & Singh, S. (in preparation). Between MDPs and Semi-MDPs: learning, planning, and representing knowledge at multiple temporal scales. *Journal of AI Research*.

Thrun, S. & Schwartz, A. (1995). Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7* (pp. 385–392). MIT Press.