Multi-Time Models for Reinforcement Learning

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

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

Abstract

Reinforcement learning can be used not only to predict rewards, but also to predict states, i.e. to learn a model of the world's dynamics. Models can be defined at different levels of temporal abstraction. Multi-time models are models that focus on predicting what will happen, rather than when a certain event will take place. Based on multi-time models, we can define abstract actions, which enable planning (presumably in a more efficient way) at various levels of abstraction.

1 Action models

Model-based reinforcement learning offers a possible solution to the problem of integrating planning in a real-time learning agent. Models are used to make predictions about the environment. The input of a model is a state and an action. The model should output a distribution of possible future states, as well as the expected value of the reward along the way. Since models provide action-dependent predictions, they also define policies for achieving certain states, or for achieving maximum expected rewards.

Reinforcement learning algorithms have traditionally been concerned with 1-step models, which assume that the agent interacts with its environment at some discrete, lowest-level time scale. We extend this framework by defining multi-time models, which describe the environment at different time scales.

Multi-time models are a formalism for describing abstract actions. Intuitively, abstract actions should express complex behaviors, which can take variable amounts of time. Our main goal is to achieve faster learning and cheaper planning by using abstract actions in addition to primitive actions. As a consequence, the models describing abstract actions should be suitable for planning (i.e. should allow the development of "good" policies), expressive and learnable.

2 Value functions

A key step in many reinforcement learning systems is the computation of state value functions. The value function for a policy π associates to each state s the expected value of the total discounted reward that the agent gathers when starting in s and acting according to π:

$$V^\pi(s) = E_\pi\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\},$$

where $\gamma \in [0,1)$ is the discount-rate parameter. The optimal value function V^* associates to each state s the maximum expected reward that can be achieved when starting in s:

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

An agent that acts greedily with respect to V^* will maximize its expected reward. Since value functions are the basis of planning in reinforcement learning systems, action models should enable us to accurately compute these functions.

Conventionally, in reinforcement learning, we assume that the agent interacts with its environment at some discrete, lowest-level time scale $t = 0, 1, 2, \ldots$ We will also make a discrete states and actions assumption: at each time step, the agent is in one of m states $s_t \in \{1, 2, \ldots, m\}$ and can choose one action a_t from a set $\mathcal{P}$ of primitive actions. In response to a_t, the environment produces a reward r_{t+1} and the state changes to s_{t+1}.

In this paper, we will use a vector representation of the states. Each state s will have a corresponding m-dimensional unit basis vector $\mathbf{s}$. All the components of $\mathbf{s}$ are 0, except the one corresponding to s, which is 1. For example, state $s = 2$ is represented through the column vector

$$\mathbf{s} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

We denote by $\mathbf{S}$ the set of all state vectors (i.e., the unit-basis vectors of length m).

The value function of a policy π can also be represented as an m-vector $\mathbf{v}^\pi$ such that ${\mathbf{v}^\pi}^T\mathbf{s} = V^\pi(s), \forall \mathbf{s} \in \mathbf{S}$ (where $\mathbf{x}^T$ denotes the transpose of vector $\mathbf{x}$). Similarly, the optimal value function V^* will be represented thourgh the m-vector $\mathbf{v}^*$.

3 1-step models

The model for a primitive action a consists of an m-vector $\mathbf{r}_a$ and an $m \times m$ matrix P_a. The vector $\mathbf{r}_a$ contains the expected immediate rewards for taking action a in each state. Thus:

$$\mathbf{r}_a^T\mathbf{s} = E\{r_{t+1}|\mathbf{s}_t = \mathbf{s}, a_t = a\}, \forall \mathbf{s} \in \mathbf{S}$$

The components of P_a are γp_{ij}^a, where $p_{ij}^a = Pr\{s_{t+1} = j | s_t = i, a_t = a\}$ is the probability of the transition from state i to state j, given action a. Thus, for any state vector $\mathbf{s}$,

$$P_a^T\mathbf{s} = \gamma E\{\mathbf{s}_{t+1}|\mathbf{s}_t = \mathbf{s}, a_t = a\}, \forall \mathbf{s} \in \mathbf{S}$$

We call the pair $\mathbf{r}_a, P_a$ the 1-step model for action a. Together, the 1-step models of all primitive actions completely characterize the dynamics of the environment.

We define 1-step models for a policy as the 1-step models of the actions taken under that policy, averaging over action models in the case of stochastic policies. Thus, the 1-step model $\mathbf{r}_\pi, P_\pi$ of a policy π satisfies the following relations:

$$\mathbf{r}_\pi^T\mathbf{s} = E_\pi\{r_{t+1}|\mathbf{s}_t = \mathbf{s}\},$$
$$P_\pi^T\mathbf{s} = \gamma E_\pi\{\mathbf{s}_{t+1}|\mathbf{s}_t = \mathbf{s}\}, \forall \mathbf{s} \in \mathbf{S}.$$

4 Planning in reinforcement learning

1-step models can be used to compute value functions through the Bellman equations for prediction and control. These equations can be written in vector form using the notation introduced above.

The Bellman equation for a policy π has the form

$$\mathbf{v}^\pi = \mathbf{r}^\pi + P_\pi \mathbf{v}^\pi, \tag{1}$$

where $\mathbf{r}^\pi, P_\pi$ are the 1-step model for π.

Similarly, the Bellman optimality equation can be expressed as

$$\mathbf{v}^* = \max_{a \in \mathcal{P}}\{\mathbf{r}_a + P_a\mathbf{v}^*\}, \tag{2}$$

where $\mathbf{v}^*$ is the m-vector containing the optimal state values, the max function is applied component-wise, and spans over the set of primitive actions $\mathcal{P}$.

The Bellman equations are usually used to derive update rules that compute $\mathbf{v}^\pi$ and $\mathbf{v}^*$. The computation of $\mathbf{v}^\pi$ starts with an arbitrary initial approximation $\mathbf{v}_0^\pi$, and performs at each step an update of the form:

$$\mathbf{v}_{k+1}^\pi \leftarrow \mathbf{r}_\pi + P_\pi \mathbf{v}_k^\pi$$

Similarly, $\mathbf{v}^*$ can be computed by iteratively applying the following update rule:

$$\mathbf{v}_{k+1}^* \leftarrow \max_{a \in \mathcal{P}}\{\mathbf{r}_a + P_a\mathbf{v}_k^*\}$$

We want to reverse the roles and use the value functions in order to define models that are suitable for planning. A model is considered suitable for planning if and only if it helps to compute the value functions. In particular, a suitable model has to satisfy the Bellman equations. The following two sections introduce two notions that help define suitable models.

5 Valid models

Definition 1 A model $\mathbf{r}, P$, is *valid for policy* π if and only $\lim_{k\to\infty} P^k = 0$, and

$$\mathbf{v}^\pi = \mathbf{r} + P\mathbf{v}^\pi. \tag{3}$$

Any valid model can be used to compute $\mathbf{v}^\pi$ via the iteration algorithm $\mathbf{v}_{k+1} \leftarrow \mathbf{r} + P\mathbf{v}_k$. This is a direct sense in which the validity of a model implies that it is suitable for planning. The simplest model satisfying this property is the 1-step model of π. We want to identify other models with the same property.

For some purposes, it is more convenient to write a model $\mathbf{r}, P$ as a single $(m+1) \times (m+1)$ matrix:

$$M = \left[\begin{array}{c|c} 1 & - \; 0 \; - \\ \hline | & \\ \mathbf{r} & P \\ | & \end{array}\right].$$

We say that the model M has been put in homogeneous coordinates. The vectors corresponding to the value functions can also be put into homogeneous coordinates, by adding an initial element that is always 1.

With this notation, the generalized Bellman equation becomes:

$$\mathbf{v}^\pi = M\mathbf{v}^\pi \tag{4}$$

Homogeneous coordinates make it easy to prove the following closure properties of valid models.

Theorem 1 *Closure under composition.* Let M_1 and M_2 be two arbitrary models that are valid for some policy π. Then the composed model M_1M_2 is also valid for π.

Proof: $(M_1M_2)\mathbf{v}^\pi = M_1(M_2\mathbf{v}^\pi) = M_1\mathbf{v}^\pi = \mathbf{v}^\pi$ ∎

Theorem 2 *Closure under averaging.* For any set of models $\{M_j\}$ that are valid for policy π and any set of diagonal matrices $\{D_j\}$ such that $\sum_j D_j = I$, the weighted model $\sum_j D_jM_j$ is also valid for π.

Proof: $(\sum_j D_jM_j)\mathbf{v}^\pi = \sum_j D_j(M_j\mathbf{v}^\pi) = \sum_j D_j\mathbf{v}^\pi = I\mathbf{v}^\pi = \mathbf{v}^\pi$ ∎

Using composition and averaging we can create a large space of valid models for a policy, starting from the 1-step model.

6 Non-overpromising models

We now turn to the full reinforcement learning problem, including control. We would like to define abstract actions that can be used in the Bellman optimality equation (2), just like a primitive action. In other words, we would like to write a generalized Bellman optimality equation:

$$\mathbf{v}^* = \max_{a\in\mathcal{P}\cup\mathcal{A}}\{\mathbf{r}_a + P_a\mathbf{v}^*\}, \tag{5}$$

in which $\mathcal{A}$ is the set of abstract actions. We want the abstract actions to help us compute $\mathbf{v}^*$ faster but, of course, we do not want them to change the value of $\mathbf{v}^*$. An abstract action will not affect the optimal value function, as long as its associated model $\mathbf{r}, P$ does not predict more reward than it is really possible to achieve. This property is captured in the following definition.

Definition 2 A model $\mathbf{r}, P$, is called *non-overpromising (NOP)* if and only if P has only positive elements, $\lim_{k\to\infty} P^k = 0$, and

$$\mathbf{v}^* \geq \mathbf{r} + P\mathbf{v}^*, \tag{6}$$

where the $\geq$ relation has to be true component-wise.

Equivalently, in homogeneous coordinates, for a non-overpromising model, M, we have:

$$\mathbf{v}^* \geq M\mathbf{v}^*. \tag{7}$$

There are closure properties for NOP models, similar to the ones for valid models. To prove these properties, we will use the following lemma:

Lemma 1 Let $M = \mathbf{r}, P$ be any non-overpromising model and $\mathbf{v}_1$, $\mathbf{v}_2$ be two vectors such that $\mathbf{v}_1 \leq \mathbf{v}_2$. Then $M\mathbf{v}_1 \leq M\mathbf{v}_2$.

Proof: If $\mathbf{v}_1 \leq \mathbf{v}_2$ then, for any matrix P with positive elements, $P\mathbf{v}_1 \leq P\mathbf{v}_2$ (by weighted summation of the initial set of inequalities. Therefore, $\mathbf{r}+P\mathbf{v}_1 \leq \mathbf{r}+P\mathbf{v}_2$ or, equivalently, $M\mathbf{v}_1 \leq M\mathbf{v}_2$ ∎

Theorem 3 *Closure under composition.* If M_1 and M_2 are non-overpromising models, the composed model M_1M_2 is also non-overpromising.

Proof: $(M_1M_2)\mathbf{v}^* = M_1(M_2\mathbf{v}^*) \leq M_1\mathbf{v}^* \leq \mathbf{v}^*$ (using lemma 1 and the hypothesis) ∎

Remark 1 If the models M_1 and M_2 correspond to the actions a_1 and a_2, the composed model M_1M_2 corresponds to an abstract action that would execute a_1 followed by a_2.

Theorem 4 *Closure under column-wise averaging.* For any set of non-overpromising models $\{M_j\}$ and any set of diagonal weighting matrices $\{D_j\}$ with positive elements such that $\sum_j D_j = I$, the weighted model $\sum_j D_jM_j$ is also non-overpormissing.

Proof: $(\sum_j D_jM_j)\mathbf{v}^* = \sum_j D_j(M_j\mathbf{v}^*) \leq \sum_j D_j\mathbf{v}^* = I\mathbf{v}^* = \mathbf{v}^*$ (using lemma 1 for D_j) ∎

Remark 2 If the models $\{M_j\}$ correspond to the abstract actions $\{a_j\}$, then the averaged model $\sum_j D_jM_j$ can be viewed as corresponding to the action of selecting among the actions $\{a_j\}$ with probabilities given by $\{D_j\}$.

The 1-step models of primitive actions are obviously non-overpromising, due to (2). We now prove the same property for 1-step policy models.

Theorem 5 For any policy π, the 1-step model M_π is non-overpromising.

Proof: Policy π is fully described by the vectors $\pi_a, a \in \mathcal{P}$, where $\pi_a(i)$ is the probability of taking the primitive action a in state i when acting according to π. Thus, $\mathbf{r}_\pi + P_\pi \mathbf{v}^* = \sum_a \pi_a^T(\mathbf{r}_a + P_a \mathbf{v}^*) \leq \sum_a \pi_a^T \mathbf{v}^* = I\mathbf{v}^* = \mathbf{v}^*$ (for the inequality, we use the fact that $\pi_a(i) \geq 0, \forall i \in \{1, \ldots, m\}$) $\blacksquare$

Remark 3 The previous proof does not assume that π is a stationary policy.

Remark 4 The non-overpromising condition for M_π can be rewritten using the fact that M_π is valid for π:

$$\mathbf{v}^* - \mathbf{v}^\pi \geq P_\pi(\mathbf{v}^* - \mathbf{v}^\pi)$$

7 The relationship between valid and non-overpromising models

Definitions 1 and 2 illustrate two different aspects that characterize "adequacy for planning". But what is the relationship between the sets of models satisfying these two conditions? A summary of this relationship is presented in figure 1.

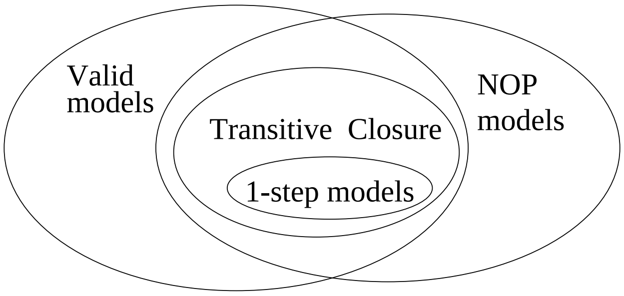

Figure 1: Relation between valid and NOP models

It is quite clear that there are non-overpromising models which are not valid for any policy π. In particular, any hopelessly pessimistic model is non-overpromising, but it cannot be achieved by any policy. Thus, the class of non-overpromising models is not included in the class of models that are valid for some policy π.

However, the fact that the set of models that are valid for some policy π is not included in the set of non-overpromising models is not at all obvious. Intuitively, it would seem that if a model is valid for some policy π, it should also be non-overpromising. In fact, this is not true.

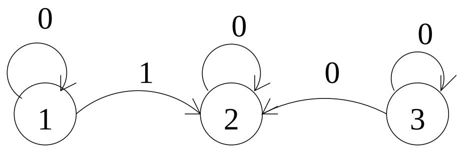

Figure 2: A Markov chain example, to illustrate the fact that not all valid models are non-overpromising

An illustration can be provided by considering the Markov chain in figure 2. The environment has three possible states, and there are two possible actions: a_1 which preserves the same state, without giving any reward, and a_2 which always causes a transition to state 2. All the transitions are deterministic, and the immediate rewards are as shown on the arrows in figure 2.

With a discount factor of γ, the models of the two primitive actions can be written as:

$$\mathbf{r}_{a_1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \; P_{a_1} = \begin{bmatrix} \gamma & 0 & 0 \\ 0 & \gamma & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

$$\mathbf{r}_{a_2} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \; P_{a_2} = \begin{bmatrix} 0 & \gamma & 0 \\ 0 & \gamma & 0 \\ 0 & \gamma & 0 \end{bmatrix}$$

The optimal value function for this environment can be easily computed:

$$\mathbf{v}^* = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Let us consider a policy π that chooses a_1 and a_2 with equal probabilities. The value function of this policy can be computed by using the prediction Bellman equation (1):

$$\begin{aligned} \begin{bmatrix} v^\pi(1) \\ v^\pi(2) \\ v^\pi(3) \end{bmatrix} &= \frac{1}{2}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \gamma & 0 & 0 \\ 0 & \gamma & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} v^\pi(1) \\ v^\pi(2) \\ v^\pi(3) \end{bmatrix} \right) \\ &+ \frac{1}{2}\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & \gamma & 0 \\ 0 & \gamma & 0 \\ 0 & \gamma & 0 \end{bmatrix} \begin{bmatrix} v^\pi(1) \\ v^\pi(2) \\ v^\pi(3) \end{bmatrix} \right) \\ &= \begin{bmatrix} \frac{\gamma}{2}v^\pi(1) + \frac{1}{2} + \frac{\gamma}{2}v^\pi(2) \\ \gamma v^\pi(2) \\ \frac{\gamma}{2}v^\pi(3) + \frac{\gamma}{2}v^\pi(2) \end{bmatrix} \end{aligned}$$

The resulting values are:

$$\begin{bmatrix} v^\pi(1) \\ v^\pi(2) \\ v^\pi(3) \end{bmatrix} = \begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ 0 \end{bmatrix}$$

Let us consider a model with the following P matrix:

$$\begin{bmatrix} 0 & \gamma & 0 \\ 0 & \gamma & 0 \\ \gamma & 0 & 0 \end{bmatrix}$$

The interresting aspect of this model is that it predicts a transition from state 3 to state 1, which in fact is not possible. We compute the reward vector for the model in such a way as to make it valid for policy π described above:

$$\begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} + \begin{bmatrix} 0 & \gamma & 0 \\ 0 & \gamma & 0 \\ \gamma & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ 0 \end{bmatrix},$$

which yields

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ -\frac{\gamma}{2-\gamma} \end{bmatrix}.$$

We now check to see if the model $\mathbf{r}, P$ that we defined is non-overpromising:

$$\begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ -\frac{\gamma}{2-\gamma} \end{bmatrix} + \begin{bmatrix} 0 & \gamma & 0 \\ 0 & \gamma & 0 \\ \gamma & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ \gamma\frac{1-\gamma}{2-\gamma} \end{bmatrix}$$

The optimal value predicted by this model for state 3 is strictly positive, so it exceeds the true optimal value $v^*(3) = 0$. The model $\mathbf{r}, P$ is valid, but overpromising.

The model $\mathbf{r}, P$ has another interresting property: it cannot be generated through any sequence of compositions and averaging operations applied to the 1-step model of the environment. Intuitively, one would expect that all the models that are non-overpromising and valid for some policy π belong to the transitive closure that can be built using the 1-step model of the environment as a seed and applying composition and averaging. This hypothesis is also false.

Let us consider the Markov process in figure 3. The states and transitions are the same as in the previous example, but the reward structure is changed: the agent receives a reward of 1 for switching from state 3 to state 2. Thus the optimal value function in this case is:

$$\mathbf{v}^* = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

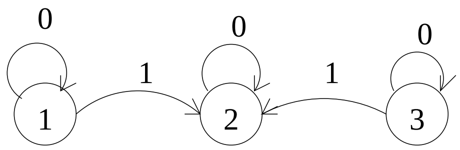

Figure 3: A Markov chain example, to illustrate that there are valid and non-overpromising models that are not in the transitive closure of 1-step models

Following the same steps as in the previous example, we can compute the value function for the policy π which chooses actions a_1 and a_2 with equal probability:

$$\mathbf{v}^\pi = \begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ \frac{1}{2-\gamma} \end{bmatrix}$$

Considering the same matrix P, we compute a new reward vector $\mathbf{r}$ such that the model $\mathbf{r}, P$ is valid for π:

$$\mathbf{r} = \begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ \frac{1-\gamma}{2-\gamma} \end{bmatrix}$$

We can now check if this model is overpromising:

$$\begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ \frac{1-\gamma}{2-\gamma} \end{bmatrix} + \begin{bmatrix} 0 & \gamma & 0 \\ 0 & \gamma & 0 \\ \gamma & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2-\gamma} \\ 0 \\ \frac{1-\gamma}{2-\gamma} + \gamma \end{bmatrix}$$

It is easy to see that, for $\gamma < 1$, we have $\frac{1}{2-\gamma} < 1$ and $\frac{1-\gamma}{2-\gamma} + \gamma < 1$, which means that $\mathbf{r}, P$ is a non-overpromising model.

8 β-models

The theory presented so far defines more precisely the conditions under which models are considered suitable for planning. The goal of defining expressive and learnable models lead to the definition of β-models (Sutton, 1995). β-models are a particular class of models, which "summarize" a mixture of models from different time scales in a single matrix. This section defines β-models and discusses their properties.

Definition 3 For any policy π having the 1-step model M_π, we call M_π^n the *n-step model of* π.

The n-step models of a policy π are built by composition from the 1-step model of the policy. Therefore, they are both valid for policy π (by theorem 1) and non-overpromising (by theorem 3).

Definition 4 *β-models.* Let π be a policy and B be a diagonal matrix that associates a parameter $\beta(i) \in [0,1]$ to each state of the environment $i \in \{1, \ldots, m\}$. Let $\beta_t = \beta(\mathbf{s}_t)$ be the parameter applicable at time step t. Then the β-model $M_\pi^B = \mathbf{r}_\pi^B, P_\pi^B$ associated to π and B is defined by:

$$\mathbf{r}_\pi^{B\,T}\mathbf{s} = E_\pi\left[\sum_{t=1}^{\infty}\gamma^{t-1}\prod_{i=1}^{t-1}\beta_i r_t \mid \mathbf{s}_0 = \mathbf{s}\right]$$

$$P_\pi^{B\,T}\mathbf{s} = E_\pi\left[\sum_{t=1}^{\infty}\gamma^{t}(1-\beta_t)\prod_{i=1}^{t-1}\beta_i \mathbf{s}_t \mid \mathbf{s}_0 = \mathbf{s}\right]$$

β_i is a measure of the importance given to the i-th state of the trajectory. In particular, if $\beta_i = 1$, then state i has no weight associated to it. If $\beta_i = 0$, all the remaining weight along the trajectory is given to state i; thus, state i is an "outcome" state. In general, all states with $\beta_i < 1$ can be viewed as "outcome" states, for which $1 - \beta_i$ is the probability of stopping the current trajectory in state i, and β_i is the probability of continuing the trajectory. We are currently working on proving that β-models are valid and non-overpromissing.

Although we have defined β-models based on 1-step models, this is not the way in which we plan to derive them. The goal is to learn β-models from experience, starting from "seeds", without having to go through 1-step models.

A possible seed that one can use is the policy π together with the matrix of β values B. Sutton (1995) presents a TD-style learning algorithm for β-models. In essence, the algorithm applies at each time step t the following corrections:

$$\Delta\mathbf{r}_t = \alpha[r_{t+1} + \mathbf{r}_t^T(\gamma\beta_{t+1}\mathbf{s}_{t+1} - \mathbf{s}_t)]\mathbf{e}_{t+1}$$

$$\Delta P_t = \alpha[(1-\beta_{t+1})\gamma\mathbf{s}_{t+1} + P_t^T(\gamma\beta_{t+1}\mathbf{s}_{t+1} - \mathbf{s}_t)]\mathbf{e}_{t+1}^T,$$

where α is the step size, and $\mathbf{e}_{t+1}$ is the m-vector containing the eligibility traces of all the states. The eligibility traces are updated using the following rule:

$$\mathbf{e}_{t+1} \leftarrow \gamma\lambda\beta_t\mathbf{e}_t + \mathbf{s}_t$$

This learning algorithm is completely on-line and incremental, and its complexity is comparable to that of regular 1-step TD-learning. Its major disadvantage is that it needs a whole specification of policy π, which is costly to get in general. Besides, our purpose is to learn policies that are somehow useful, rather that starting with predefined ones.

This is why we consider that a better starting seed would be to specify the outcome states (i.e. the B matrix) along with their hypothetical values U. In this setting, we basicaly suggest that the agent learns policies for achieving certain outcome states. Q-learning or any other reinforcement learning algorithm can then be applied to learn the optimal state-action value function $Q^*_{U,B}$. Policy π is then defined by acting greedily with respect to $Q^*_{U,B}$, and we can learn the corresponding β-model as in the previous case.

An alternative to specifying hypothetical values for outcome states would be to overlay additional rewards over those provided by the environment, in order to bias the system towards certain actions. However, given the semantics of β-models, the previous alternative seems preferable.

9 Open Questions

There are several open questions that need to be investigated:

- Can we characterize the intersection between the set of non-overpromising models and the set of valid models in a unified way?
- Outside the transitive closure of 1-step models, how can we capture the difference between the valid models that are non-overpromising and those that are not?
- Can we learn Q-values for abstract actions? The major difference between abstract and primitive action is that abstract action can take indefinite amounts of time. Thus, the agent should be able to redecide on its choice of action before the current action is complete. In this case, the action that didn't finish should be assigned partial credit, based on the value of the state in which the action was interrupted.
- Can models be learned in parallel with learning the optimal value function?
- Instead of providing the β-values to the system, could we learn them from experience as well?
- What is the merit of building new abstract actions from existing ones using composition and averaging?
- How can we establish which abstract actions are useful, in order to "prune" the space of actions we can choose from?

Acknowledgments

The authors thank Paul Utgoff, Satinder Singh, Peter Dayan, Gunnar Blix and the members of the ANW lab at the University of Massachusetts for encouragement, ideas, and comments contributing to this paper. This research was supported by NSF Grant ECS-9511805 to Andrew G. Barto and Richard S. Sutton. Doina Precup also acknowledges the support of the Fulbright Foundation.

References

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