TD models: Modeling the world at a mixture of time scales

Tianyu Li

March 2017

1 Introduction to multi-scale planning and modeling

The traditional model-based RL only assumes a single fixed time step, i.e. usually we take one action and then we observe the environment and obtain certain reward. However, some of the other tasks may not be so straightforward. For example, if we are trying to get to work, then it involves planning at both high level and low level. For the high level it could be by which means to travel to work, while low level could be when to hit the brake. We need to find a way somehow simultaneously optimizing both the low level and the high level dynamics to maintain our general objective and make every step accurate.

2 Notation and generalized Bellman equation

In the paper, the author focus only on the primal problem in RL, which is to estimate the value function V^π for a fixed policy π and this estimation will only base on states and reward, the actions have been ignored. The followings are some of the notations: we denote the state reward sequence as $s_0, r_0, s_1, r_1, \cdots$, where $s_i \in S, r_i \in R$, and the state transition probabilities as $p(i,j) = E(s_{t+1} = j | s_t = i)$ and the $m \times m$ matrix of such probabilities as P. Denote the state vector as x_t, for which all the components are zeros, except in the tth slot, it will be one. Hence the value function can be written as:

$$V^\pi(s_t) = \sum_{k=0}^{\infty} \gamma^k R^T P^k x_t \tag{1}$$

The Bellman equation can thus be written as:

$$V = R + \gamma P^T V \tag{2}$$

Let us replace R with $\mathcal{R}$ and γP with $\mathcal{P}$, then for any $\mathcal{P}$ and $\mathcal{R}$ that satisfies the Bellman equation $V = \mathcal{R} + \mathcal{P}^T V$ are said to be a valid model. Moreover, we can construct a look-ahead updating rule for this equation:

$$V_{t+1} = \mathcal{R} + \mathcal{P}^T V_t \tag{3}$$

3 N-step models

The most common extension of the one-step look-ahead model is by expanding the original Bellman equation once more, i.e.:

$$V = R + \gamma P^T R + (\gamma P^T)^2 R \tag{4}$$

Then by replacing $R+\gamma P^T R$ with $\mathcal{R}$ and $(\gamma P^T)^2$ with $\mathcal{P}$, we have $V = \mathcal{R}+\mathcal{P}^T V$. As one can imagine, we can just set $\mathcal{R}^n = \sum_{k=0}^{n-1}(\gamma P^T)^k R$ and $\mathcal{P}^{(n)} = (\gamma P)^n$ we can then obtain the n-step model.

With the n-step model and k-look-ahead updating, people can generate various of settings and it has been shown that whatever the combination is, the end result-convergence to V- is not affected. However, the two major drawbacks of the n-step model is that the n is fixed and for large n it might be really expensive to learn.

4 Intermixing time scales

Although directly predicting n-step ahead could be straightforward, but the result time series might not be as smooth as it should be. Hence a temporal prediction might be more desirable. Now, instead of directly predicting x_{t+10}, we predict a weighted average of states in the neighborhood of x_t. To formulate this idea, let w_n denote the weight given in the weighted average of the n-step prediction, and we have $\sum_{n=1} w_n = 1$. Then the multi-scale model should be: $\mathcal{P} = \sum_{n=1} w_n \mathcal{P}^{(n)}$ and $\mathcal{R} = \sum_{n=1} w_n \mathcal{R}^{(n)}$, we will also refer to this model as 'blurred' n-step mixture. An interesting fact about this model is that it has been shown that any unitary mixture of n-step models is a valid mode. This means that by using the 'blurred' n-step mixture, not only can we achieve any linear combination of n-step and k-look-ahead, we can also gurranteen that this will still converge to V.

5 β model

Now let us consider how to determine the value of the weights w_n. As a matter of fact, we want the weight to decline exponentially with the decay, thus we set w_n with a $0 < \beta < 1$ parameter.

$$w_n = (1-\beta)\beta^{n-1} \tag{5}$$

Why don't we just learn β value from the data? For example we can just think of β as another parameter when doing TD update.

However, due to the states' dynamics, we also want to potentially assign different β to different β_i, and this result in the so called full β models. Let $\beta(i)$ denote the β assigned to state s_i, then the weight can be written as:

$$w_n = (1 - \beta_n) \prod_{i=1} \beta_i \tag{6}$$

Formally, the full β model $\mathcal{P}$ and $\mathcal{R}$ can be written as:

$$\mathcal{P} = \gamma(I - B)P \sum_{k=0} *(\gamma BP)^k \tag{7}$$

$$\mathcal{R} = \sum_{k=0} (\gamma P^T B)^k R \tag{8}$$

where B is the diagonal matrix of $\beta(i)$.

The full β models are very general models, and they can represent variable mixtures of time scales dependent on the state trajectory taken. In fact, the parameter β seems like another version of γ, in the sense that it can decide the alive-death status of the Markov Chain. For example, for the action "look for your keys", we can set β to one for the state in which one is still looking for the key, telling the chain to keep going, while we can set it to zero for the states that already found the key.

6 TD(λ) learning of β models

Let us now decompose the $\mathcal{P}x_t$ to perform an incremental updating rule of TD learning.

$$\begin{aligned} y_t &= \mathcal{P}x_t \\ &= E(\sum_{k=1}^{\infty} \gamma^k (1 - \beta_{t+k}) \prod_{i=1}^{k-1} \beta_{t+i} x_{t+k}) \\ &= E(\sum_{k=1}^{n} \gamma^k (1 - \beta_{t+k}) \prod_{i=1}^{k-1} \beta_{t+i} x_{t+k} + \gamma^n \prod_{i=1}^{n} \beta_{t+i} \mathcal{P}x_{t+n}) \end{aligned} \tag{9}$$

Although we can't really work with the expectation, however we can just sample from it. Let us replace each expected state with the observed state and the final $\mathcal{P}$ with the current estimate $\mathcal{P}_t$, then we will have an n-step TD target.

$$y_t^{(n)} = \sum_{k=1}^{n} \gamma^k (1 - \beta_{t+k}) \prod_{i=1}^{k-1} \beta_{t+i} x_{t+k} + \gamma^n \prod_{i=1}^{n} \beta_{t+i} \mathcal{P}x_{t+n} \tag{10}$$

Therefore the TD update rule will be:

$$\Delta \mathcal{P}_t = \alpha(y_t^{\lambda} - \mathcal{P}_t x_t) x_t^T \tag{11}$$

Where y_t^λ involves the flavour of TD(λ) and is defined as $y_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}y_t^{(n)}$. Here α is the learning rate.

However, the major problem for this setting is that we do not know $\mathcal{P}_t$ at time stet t, because we have not observed x_{t+n}. Taking the advantage of the fact that the estimate being updated does not change greatly from time step to time step. Let us denote the TD error by ϵ_t, then it can be written as:

$$\epsilon_t = (1 - \beta_{t+1}\gamma x_{t+1} + \gamma\beta_{t+1}\mathcal{P}_t x_{t+1} - \mathcal{P}_t x_t) \tag{12}$$

and then we will have our learning rule:

$$\Delta\mathcal{P}_t = \alpha\epsilon_t \sum_{k=0}^{\infty}(\gamma\lambda)^k \prod_{i=0}^{k-1}\beta_{t-i}x_{t-k}^T \tag{13}$$

A similar learning rule for $\mathcal{R}$ is as following:

$$\Delta\mathcal{R}_t = \alpha\epsilon'_t \sum_{k=0}^{\infty}(\gamma\lambda)^k \prod_{i=0}^{k-1}\beta_{t-i}x_{t-k}^T \tag{14}$$

where $\epsilon'_t = (1 - \beta_{t+1}\gamma x_{t+1} + \gamma\beta_{t+1}\mathcal{R}_t x_{t+1} - \mathcal{R}_t x_t)$