[S&B Book] Chapter 6: Temporal-Difference Learning

Tags

- Temporal-Difference (TD) learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas
  - Like MC, TD can learn directly from raw experience without a model of the environment's dynamics;
  - Like DP, TD updates estimates based in part on other learned estimates without waiting for a final outcome (they bootstrap).

6.1 TD Prediction

- TD method:

  $$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

  The simplest TD method makes the update immediately on the transition to S_{t+1} and receiving R_{t+1};

  This method is also called ***one-step* TD, or TD(0)**, because it is a special case of the TD(λ) and n-step TD methods.

- Algorithm:

```
Tabular TD(0) for estimating $v_\pi$

Input: the policy $\pi$ to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        $A \leftarrow$ action given by $\pi$ for $S$
        Take action $A$, observe $R$, $S'$
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until $S$ is terminal
```

- *sample updates* and *expected updates*
    - TD and MC are ***sample updates*** because they involve looking ahead to a sample successor state (or state-action pair), using the value of successor and the reward along the way to compute a back-up value, and then updating the value of the original state (or state-action pair) accordingly;
    - DP methods are ***expected updates*** because they are based on complete distribution of all possible successors;
- **TD error:**

  The difference between the estimated value of S_t and the better estimate $R_{t+1} + \gamma V(S_{t+1})$; the TD error arises in various forms throughout reinforcement learning;

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

- If the array V does not change during the episode (as it does not change in Monte Carlo methods), then the Monte Carlo error can be written as a sum of TD errors:

$$\begin{aligned} G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S) + \gamma V(S_{t+1}) + -\gamma V(S_{t+1}) \\ &= \delta_t + \gamma(G_{t+1} - V(S_{t+1})) \\ &= \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{T-t-1}\delta_{T-1} + \gamma^{T-t}(G_T - V(S_T)) \\ &= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k \end{aligned}$$

Exercise 6.1:

If V changes during the episode, then (6.6) only holds approximately; what would the difference be between the two sides? Let Vt denote the array of state values used at time t in the TD error (6.5) and in the TD update (6.2). Redo the derivation above to determine the additional amount that must be added to the sum of TD errors in order to equal the Monte Carlo error.

Solution:

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

$$\begin{aligned} G_t - V_t(S_t) &= R_{t+1} + \gamma G_{t+1} - V_t(S_t) + \gamma V_t(S_{t+1}) - \gamma V_t(S_{t+1}) \\ &= \delta_t + \gamma\left(G_{t+1} - V_t(S_{t+1})\right) \\ &= \delta_t + \gamma(G_{t+1} - V_{t+1}(S_{t+1}) - \alpha[R_t + \gamma V_t(S_{t+1}) - V_t(S_t)]) \\ &= \delta_t + \gamma(G_{t+1} - V_{t+1}(S_{t+1})) + \gamma \Delta V_{t+1}(S_t) \\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - V_{t+2}(S_{t+2})) + \gamma\Delta V_{t+1}(S_t) + \gamma^2 \Delta V_{t+2}(S_{t+1}) \\ &= \cdots \\ &= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k + \gamma^{k-t+1}\Delta V_{k+1}(S_k) \end{aligned}$$

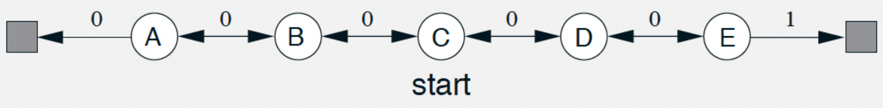
6.2 Advantages of TD Prediction Methods

1. TD methods over DP methods: **TD methods do not require a model of the environment**, of its reward and next-state probability distributions.
2. TD methods over MC methods: **TD methods are naturally implemented in an online, fully incremental fashion.** The advantage becomes obvious when:
   - Some applications have very long episodes so that delaying all learning until the end of the episode is too slow.
   - Some applications are continuing tasks and have no episodes at all.
3. TD methods over MC methods: MC methods must ignore or discount episodes on which experimental actions are taken, which can generally slow learning. **TD methods learn from each transition regardless of what subsequent actions are taken.**

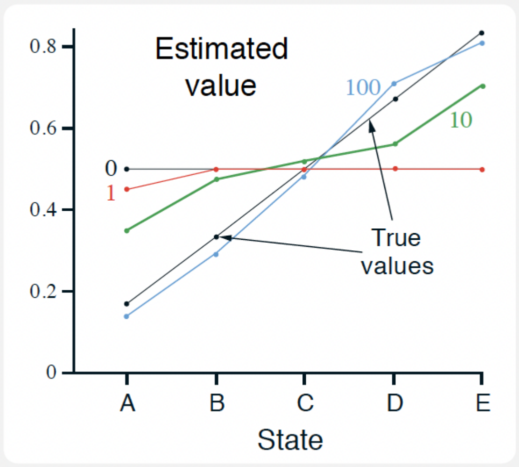
TD methods have usually been found to converge faster than constant-α MC methods on stochastic tasks as illustrated in the following example.

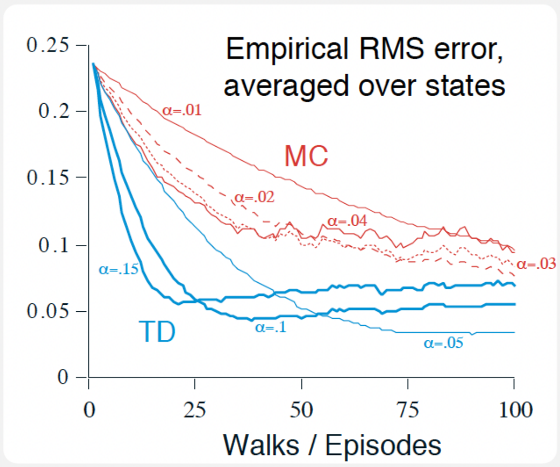
Example 6.2 Random Walk

In this example we empirically compare the prediction abilities of TD(0) and constant-α MC when applied to the following Markov reward process:

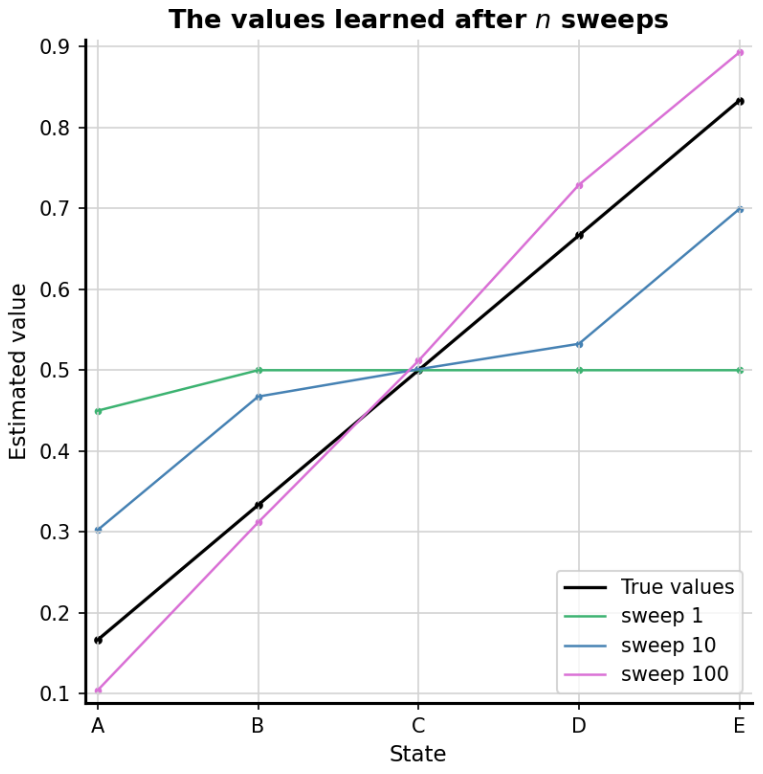


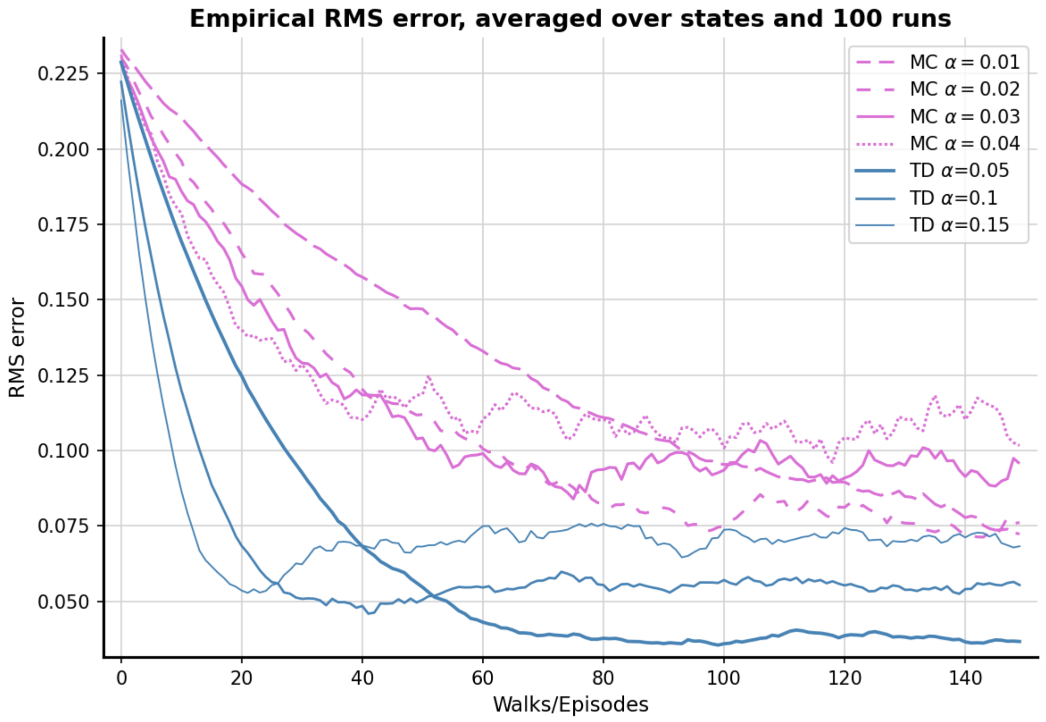
A *Markov reward process*, or MRP, is a Markov decision process without actions. We will often use MRPs when focusing on the prediction problem, in which there is no need to distinguish the dynamics due to the environment from those due to the agent. In this MRP, all episodes start in the center state, C, then proceed either left or right by one state on each step, with equal probability. Episodes terminate either on the extreme left or the extreme right. When an episode terminates on the right, a reward of +1 occurs; all other rewards are zero. For example, a typical episode might consist of the following state-and-reward sequence: $\mathsf{C}, 0, \mathsf{B}, 0, \mathsf{C}, 0, \mathsf{D}, 0, \mathsf{E}, 1$. Because this task is undiscounted, the true value of each state is the probability of terminating on the right if starting from that state. Thus, the true value of the center state is $v_\pi(\mathsf{C}) = 0.5$. The true values of all the states, A through E, are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}$, and $\frac{5}{6}$.





Implementation


The values learned after n sweeps
Estimated value
State
A
B
C
D
E
True values
sweep 1
sweep 10
sweep 100


Empirical RMS error, averaged over states and 100 runs
RMS error
Walks/Episodes
MC $\alpha = 0.01$
MC $\alpha = 0.02$
MC $\alpha = 0.03$
MC $\alpha = 0.04$
TD α=0.05
TD α=0.1
TD α=0.15

https://github.com/terrence-ou/Reinforcement-Learning-2nd-Edition-Notes-Codes/blob/main/chapter_06_temporal_difference_learning/example_6_2_random_walk.py