7.1 n-Step TD Prediction

What is the space of methods lying between Monte Carlo and TD methods? Consider estimating V^π from sample episodes generated using π. Monte Carlo methods perform a backup for each state based on the entire sequence of observed rewards from that state until the end of the episode. The backup of simple TD methods, on the other hand, is based on just the one next reward, using the value of the state one step later as a proxy for the remaining rewards. One kind of intermediate method, then, would perform a backup based on an intermediate number of rewards: more than one, but less than all of them until termination. For example, a two-step backup would be based on the first two rewards and the estimated value of the state two steps later. Similarly, we could have three-step backups, four-step backups, and so on. Figure 7.1 diagrams the spectrum of *n-step backups* for V^π, with one-step, simple TD backups on the left and up-until-termination Monte Carlo backups on the right.

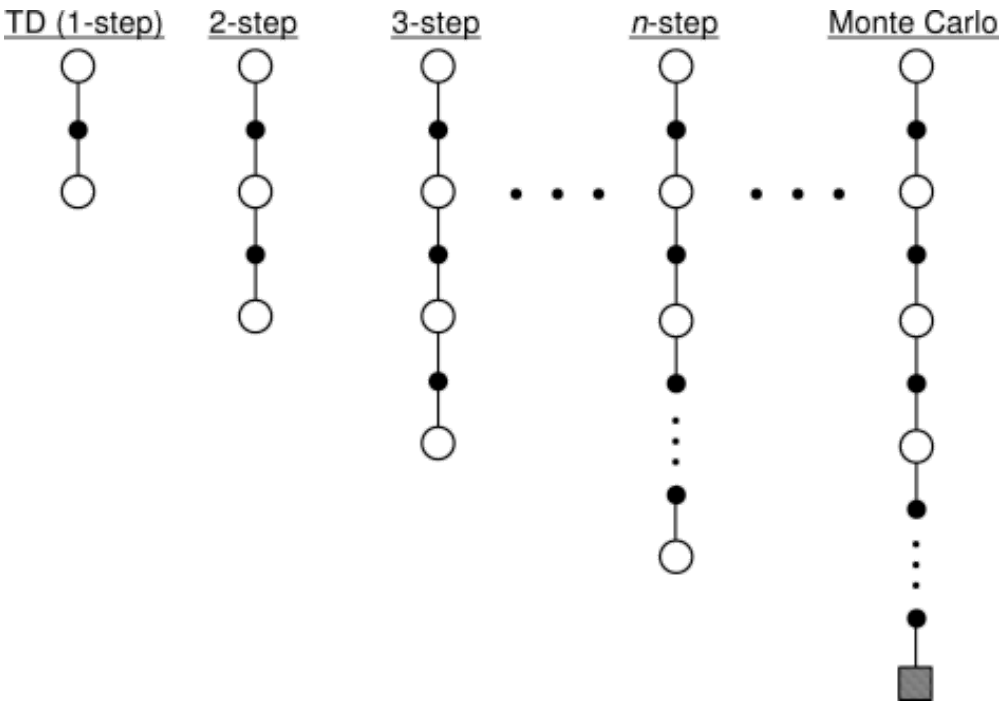

Figure 7.1: The spectrum ranging from the one-step backups of simple TD methods to the up-until-termination backups of Monte Carlo methods. In between are the n-step backups, based on n steps of real rewards and the estimated value of the nth next state, all appropriately discounted.

The methods that use n-step backups are still TD methods because they still change an earlier estimate based on how it differs from a later estimate. Now the later estimate is not one step later, but n steps later. Methods in which the temporal difference extends over n steps are called *n-step TD methods*. The TD methods introduced in the previous chapter all use one-step backups, and henceforth we call them *one-step TD methods*.

More formally, consider the backup applied to state s_t as a result of the state-reward sequence, $s_t, r_{t+1}, s_{t+1}, r_{t+2}, \ldots, r_T, s_T$ (omitting the actions for simplicity). We know that in Monte Carlo backups the estimate $V_t(s_t)$ of $V^\pi(s_t)$ is updated in the direction of the complete return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T,$$

where T is the last time step of the episode. Let us call this quantity the *target* of the backup. Whereas in Monte Carlo backups the target is the expected return, in one-step backups the target is the first reward plus the discounted estimated value of the next state:

$$R_t^{(1)} = r_{t+1} + \gamma V_t(s_{t+1}).$$

This makes sense because $\gamma V_t(s_{t+1})$ takes the place of the remaining terms $\gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$, as we discussed in the previous chapter. Our point now is that this idea makes just as much sense after two steps as it does after one. The two-step target is

$$R_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V_t(s_{t+2}),$$

where now $\gamma^2 V_t(s_{t+2})$ takes the place of the terms $\gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots + \gamma^{T-t-1} r_T$. In general, the n-step target is

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}). \tag{7.1}$$

This quantity is sometimes called the "corrected n-step truncated return" because it is a return truncated after n steps and then approximately corrected for the truncation by adding the estimated value of the nth next state. That terminology is descriptive but a bit long. We instead refer to $R_t^{(n)}$ simply as the *n-step return* at time t.

Of course, if the episode ends in less than n steps, then the truncation in an n-step return occurs at the episode's end, resulting in the conventional complete return. In other words, if $T - t < n$, then $R_t^{(n)} = R_t^{(T-t)} = R_t$. Thus, the last n n-step returns of any episode are always complete returns, and an infinite-step return is always a complete return. This definition enables us to treat Monte Carlo methods as the special case of infinite-step returns. All of this is consistent with the tricks for treating episodic and

continuing tasks equivalently that we introduced in Section 3.4. There we chose to treat the terminal state as a state that always transitions to itself with zero reward. Under this trick, all n-step returns that last up to or past termination have the same value as the complete return.

An *n-step backup* is defined to be a backup toward the n-step return. In the tabular, state-value case, the increment to $V_t(s_t)$ (the estimated value of $V^\pi(s_t)$ at time t), due to an n-step backup of s_t, is defined by

$$\Delta V_t(s_t) = \alpha \Big[R_t^{(n)} - V_t(s_t) \Big],$$

where α is a positive step-size parameter, as usual. Of course, the increments to the estimated values of the other states are $\Delta V_t(s) = 0$, for all $s \neq s_t$. We define the n-step backup in terms of an increment, rather than as a direct update rule as we did in the previous chapter, in order to distinguish two different ways of making the updates. In *on-line updating*, the updates are done during the episode, as soon as the increment is computed. In this case we have $V_{t+1}(s) = V_t(s) + \Delta V_t(s)$ for all $s \in \mathcal{S}$. This is the case considered in the previous chapter. In *off-line updating*, on the other hand, the increments are accumulated "on the side" and are not used to change value estimates until the end of the episode. In this case, $V_t(s)$ is constant within an episode, for all s. If its value in this episode is $V(s)$, then its new value in the next episode will be $V(s) + \sum_{t=0}^{T-1} \Delta V_t(s)$.

The expected value of all n-step returns is guaranteed to improve in a certain way over the current value function as an approximation to the true value function. For any V, the expected value of the n-step return using V is guaranteed to be a better estimate of V^π than V is, in a worst-state sense. That is, the worst error under the new estimate is guaranteed to be less than or equal to γ^n times the worst error under V:

$$\max_s \left| E_\pi\Big\{ R_t^{(n)} \;\Big|\; s_t = s \Big\} - V^\pi(s) \right| \;\leq\; \gamma^n \max_s \left| V(s) - V^\pi(s) \right|. \tag{7.2}$$

This is called the *error reduction property* of n-step returns. Because of the error reduction property, one can show formally that on-line and off-line TD prediction methods using n-step backups converge to the correct predictions under appropriate technical conditions. The n-step TD methods thus form a family of valid methods, with one-step TD methods and Monte Carlo methods as extreme members.

Nevertheless, n-step TD methods are rarely used because they are inconvenient to implement. Computing n-step returns requires waiting n steps to observe the resultant rewards and states. For large n, this can become problematic, particularly in control applications. The significance of n-step TD methods is primarily for theory and for understanding related methods that are more conveniently implemented. In the next few sections we use the idea of n-step TD methods to explain and justify eligibility trace methods.

Example 7.1: n-step TD Methods on the Random Walk Consider using n-step TD methods on the random walk task described in Example 6.2 and shown in Figure 6.5. Suppose the first episode progressed

directly from the center state, C, to the right, through D and E, and then terminated on the right with a return of 1. Recall that the estimated values of all the states started at an intermediate value, $V_0(s) = 0.5$. As a result of this experience, a one-step method would change only the estimate for the last state, $V(E)$, which would be incremented toward 1, the observed return. A two-step method, on the other hand, would increment the values of the two states preceding termination: $V(D)$ and $V(E)$ would both be incremented toward 1. A three-step method, or any n-step method for $n > 2$, would increment the values of all three of the visited states toward 1, all by the same amount. Which n is better? Figure 7.2 shows the results of a simple empirical assessment for a larger random walk process, with 19 states (and with a -1 outcome on the left, all values initialized to 0). Shown is the root mean-squared error in the predictions at the end of an episode, averaged over states, the first 10 episodes, and 100 repetitions of the whole experiment (the same sets of walks were used for all methods). Results are shown for on-line and off-line n-step TD methods with a range of values for n and α. Empirically, on-line methods with an intermediate value of n seem to work best on this task. This illustrates how the generalization of TD and Monte Carlo methods to n-step methods can potentially perform better than either of the two extreme methods.

■

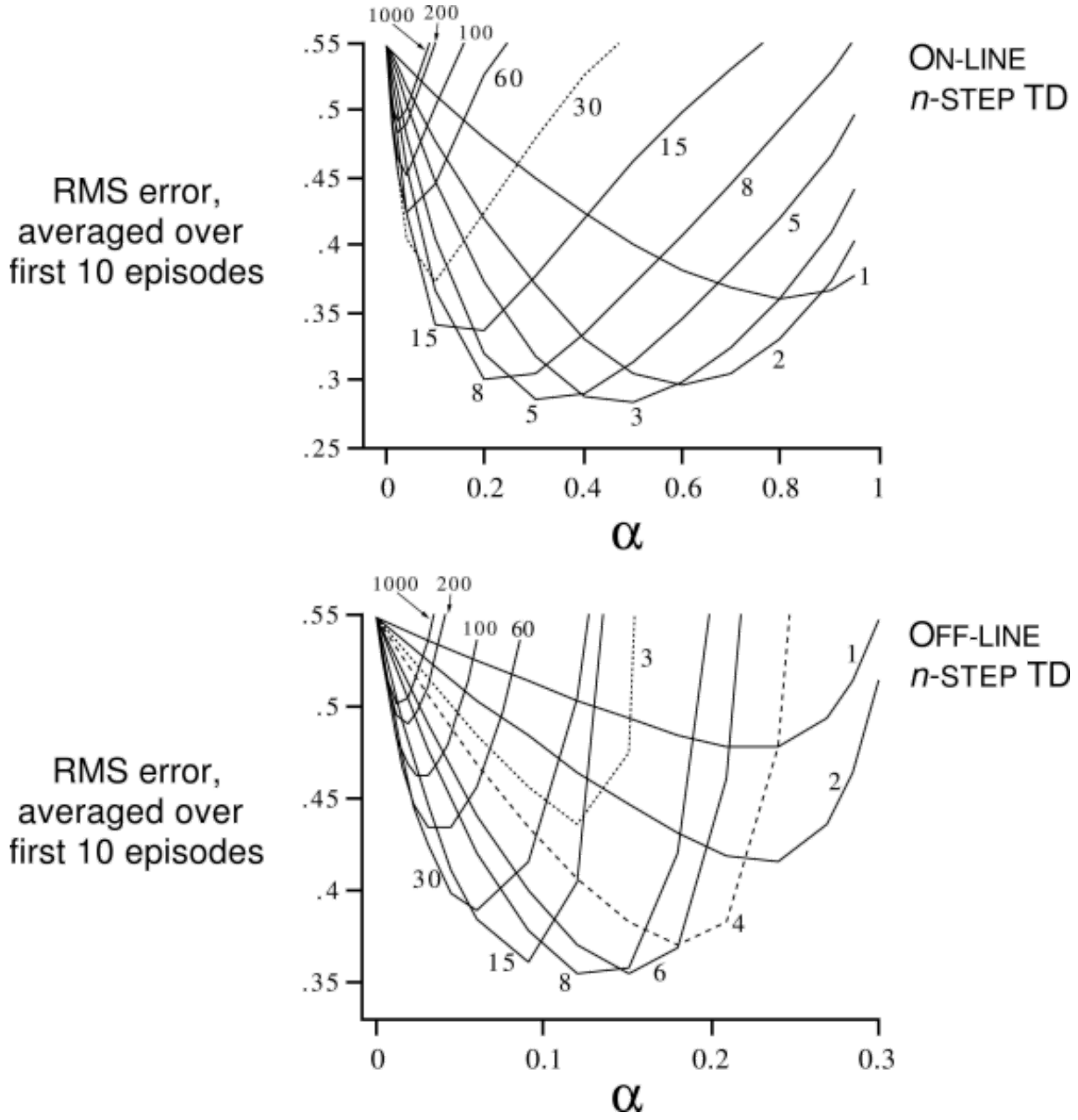

Figure 7.2: Performance of n-step TD methods as a function of α, for various values of n, on a 19-state random walk task. The performance measure shown is the root mean-squared (RMS) error between the true values of states and the values found by the learning methods, averaged over the 19 states, the first 10 trials, and 100 different sequences of walks.

Exercise 7.1 Why do you think a larger random walk task (19 states instead of 5) was used in the examples of this chapter? Would a smaller walk have shifted the advantage to a different value of n? How about the change in left-side outcome from 0 to -1? Would that have made any difference in the best value of n?

Exercise 7.2 Why do you think on-line methods worked better than off-line methods on the example task?

★**Exercise 7.3** In the lower part of Figure 7.2, notice that the plot for $n = 3$ is different from the others, dropping to low performance at a much lower value of α than similar methods. In fact, the same was observed for $n = 5$, $n = 7$, and $n = 9$. Can you explain why this might have been so? In fact, we are not

sure ourselves.

Mark Lee 2005-01-04