When does TD have lower variance than MC?

Sam Greydanus
sgrey@google.com

September 18, 2018

Abstract

The first goal of this writeup is to clarify the connection between Monte Carlo (MC) and Temporal Difference (TD) value estimation. We will use notation similar to that of Sutton and Barlow but frame the algorithms in a slightly different way.

The second goal of this writeup is to show that the expected variance of TD value estimation is always less than or equal to that of MC value estimation.

1 The MC-TD Connection

Monte Carlo (MC). Monte Carlo methods estimate the value of a state s_t by computing the average return $G(s_t)$ at that state. A simple every-visit MC update would look like Equation 1 where $\alpha = \frac{1}{k}$ and k is the visit count of s_t.

$$V(s_t) \leftarrow V(s_t) + \alpha[G(s_t) - V(s_t)] \tag{1}$$

Temporal Differences (TD). Another way to estimate the value of a state is with *temporal difference* (TD) learning. The idea of TD learning is to express the value of s_t in terms of the values of its successor state s_{t+1} along a trajectory, plus the change in reward due to the $s_t \rightarrow s_{t+1}$ transition.

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2}$$

This approach is often called *bootstrapping* because each state uses the next state's value to update its own.

Implementation notes. Looking at Equations 1 and 2, we can see that the core difference between MC and TD updates is the substitution $G(s_t) = r_{t+1} + \gamma V(s_{t+1})$. Unfortunately, the classic algorithms for MC and TD value estimation are very different. In the textbook, *Reinforcement Learning: An Introduction* [1], the MC and TD algorithms are given on pages 92 and 120 respectively. For MC learning, the authors compute $G_0 ... G_N$ across each episode and average across episodes. For TD learning, the authors update $V(s_t)$ after each state transition in an episode.

In spite of these textbook differences, most modern implementations of MC and TD are strikingly similar. The value updates happen at the end of each episode and proceed in reverse order from the last episode in the trajectory to the first. We choose to follow this format (and thus depart slightly from textbook definitions) for two reasons. First, doing so enables us to make tighter analogies between MC and TD. Second, most real-world applications of RL algorithms use this format.

Pseudocode for our versions of MC and TD value estimation can be found in Figure 1.

```
Initialize:
  $\pi \leftarrow$ policy to be evaluated
  $V_{MC}, V_{TD} \leftarrow$ initialize all state-value pairs to zero

Repeat forever:
  Generate an episode of $N$ steps using $\pi$
  For $s_t$ in Reversed$(s_1 ... s_{N-1})$ :
    $G(s_t) \leftarrow$ return at state $s_t$
    $V_{MC}(s_t) \leftarrow V_{MC}(s_t) + \alpha[G(s_t) - V_{MC}(s_t)]$
    $V_{TD}(s_t) \leftarrow V_{TD}(s_t) + \alpha[r_{t+1} + \gamma V_{TD}(s_{t+1}) - V_{TD}(s_t)]$
```

Figure 1: Estimating value using Monte Carlo and Temporal Difference methods.

2 Comparing the Variance of MC and TD Estimates

In this section, we will show why TD value estimates tend to have equal or lower variance than MC estimates. We cannot show that TD value estimates are *always* better than MC value estimates...but we can show that they are equal or better *on average*. Since both methods are unbiased estimates, this proof tells us that TD, on average, will give an equal or better value estimate than MC.

We'll start by communicating two key intuitions using examples from a 4x3 Gridworld environment. Then we'll take show why these claims are true in general.

2.1 Intuitions.

Intuition 1: how TD beats MC. Let's consider the toy example in Figure 2. In this example, an RL agent lives in a 3x4 Gridworld and has accumulated three paths of experience. Now the agent is exploring its world a fourth time, intent on earning the maximum possible reward. If it has just moved into state s_{11}, what action should it take next?

Your intuitive answer is probably "left." This is indeed the best strategy. However, Monte Carlo value estimation will assign a higher value to the "move up" action given the agent's experience. This is because when the agent was in s_{11} and it took a "move down" action it received -1 rewards (path 2). But when it took a "move up" action it earned +1 rewards (path 3). Based *only* on the paths that pass through state s_{11}, the agent should move up.

But this is a silly answer because the "move down" action takes the agent closer to the +5 reward in state s_{23}. The agent has already explored the route to the +5 reward, and it knows that $V(s_{21}, a = \text{"}right\text{"})$ is large (see the action-value estimates in the colored triangles in Figure 2). If only there was a way for the agent to access this information from state s_{11}...

Here's a solution: let the agent build synthetic paths - routes it never took - from pieces of real paths. Now the agent can merge the first part of path 2 with the second part of path 1 (Figure 3). This "synthetic" path lets the agent assign high value to the "move down" action in state s_{11}.

What we've done by merging paths 1 and 2 is let the upstream value of future states trickle backwards along paths that the agent never took. An important fact is that the agent *could* have taken these paths

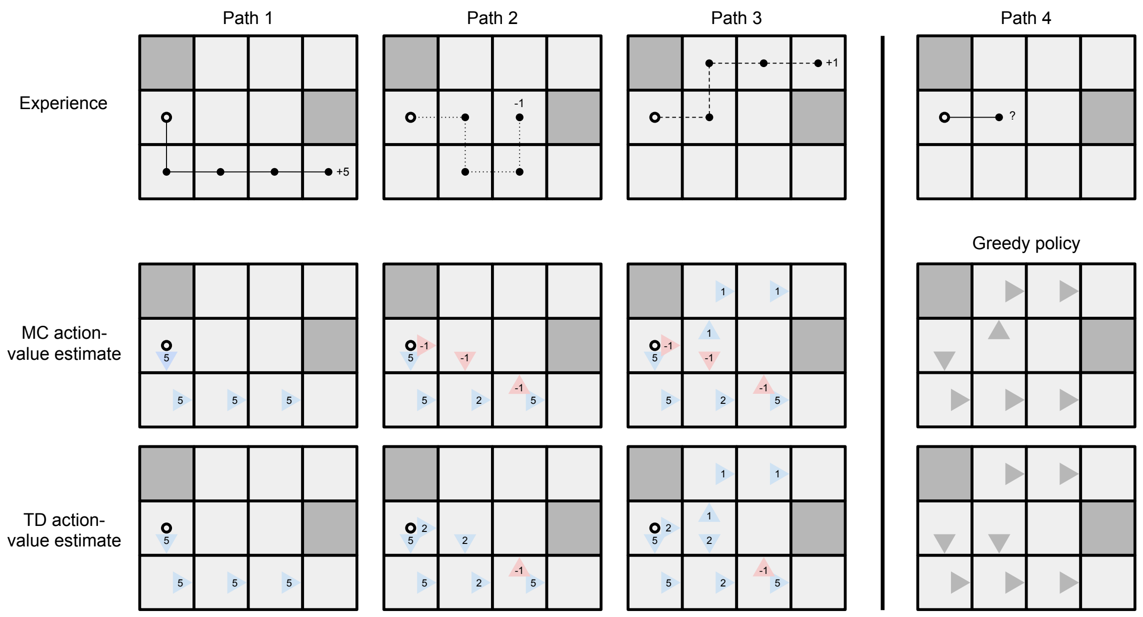

Figure 2: A 3x4 gridworld with three separate paths and the beginning of a fourth. Using its three previous paths, the agent must decide whether to take a "move up" or "move down" action. If the agent learns an action-value function using Monte Carlo estimation, it will take the less-than-optimal "move up" action whereas if it uses TD estimation, it will take the optimal "move down" action.

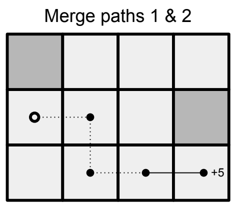

Figure 3: Combining paths 1 and 2.

because they contain only transitions that are part of the agent's experience...even if they don't occur all in the same episode.

Wherever two trajectories in an agent's experience intersect, these synthetic paths let TD learning apply outcomes from *both* the paths leading away from the intersection to *both* of the paths leading into the intersection. This is what makes TD estimates more statistically efficient than MC estimates!

Intuition 2: TD is not *always* better. Another important idea is that TD value estimates are not always better than MC value estimates. Figure 4 shows an example where the MC value estimate ends up being better.

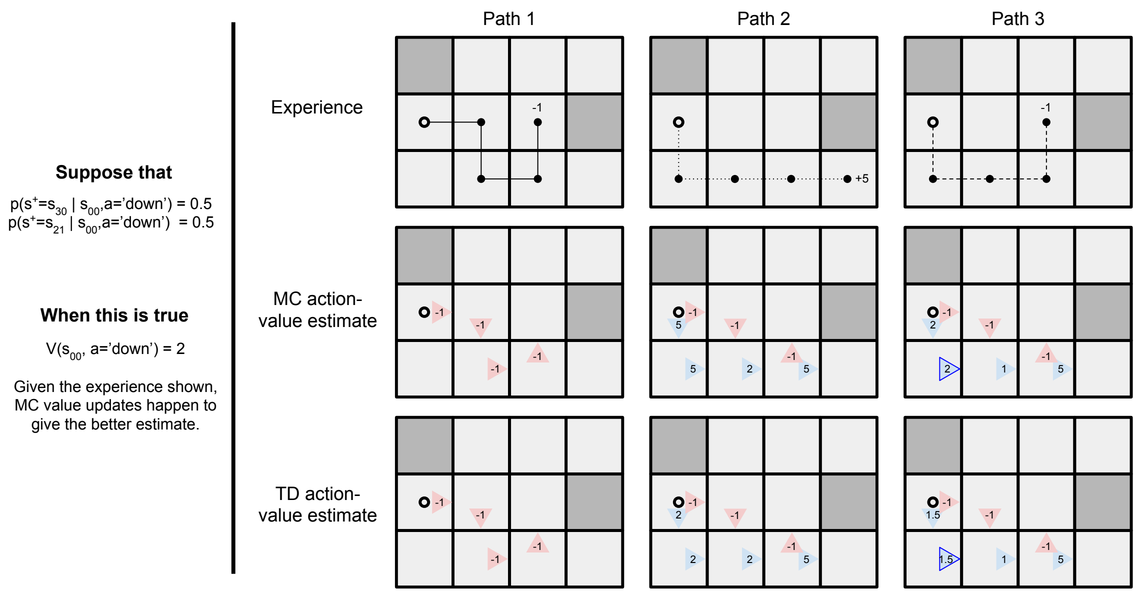
Path 1
Path 2
Path 3
Experience
MC action-value estimate
TD action-value estimate
Suppose that
p(s+=s30 | s00,a='down') = 0.5
p(s+=s21 | s00,a='down') = 0.5
When this is true
V(s00, a='down') = 2
Given the experience shown, MC value updates happen to give the better estimate.
-1
+5
5
2
1
1.5

Figure 4: TD is not always better. Here is an example where the MC value estimate for $V(s_{20}, a = \text{“}right\text{”})$ is better than the TD estimate.

Notice that we are using the same 3x4 Gridworld as we used for Intuition 1, but the agent has collected slightly different experiences. We've also assumed that, whenever the agent is in state s_{20} and takes the "move right" action, it has an equal chance of terminating the episode in s_{12} and s_{23}. From this simple assumption, we can see that $V(s_{20}, a = \text{“}right\text{”}) = 2$ in theory.

Given the three paths shown in Figure 4, the MC value estimate just happens to match the theoretical action value. Meanwhile, the TD estimate has an error of 0.5. But even though these outcomes *can* happen, they are unlikely. It ends up being more likely that the TD estimate will be better. This is why, in the proof, we'll focus on analyzing the *expected* variance of MC and TD value estimates.

2.2 Proof.

To set the scene, consider an arbitrary environment with discrete states $s_1, s_2, ...s_n$ and an agent with policy π and a value function $V^\pi(s)$. We will use Algorithm 1 to learn tabular value estimates $V^\pi_{MC}(s)$ and $V^\pi_{TD}(s)$, though we will drop the π superscript for convenience. The statement we intend to prove is:

The expected variance of a TD value estimate is always less than or equal to that of a MC value estimate. In other words, $\mathbb{E}[Var[V_{TD}(s_t)]] \leq \mathbb{E}[Var[V_{MC}(s_t)]]$ *after any number of updates.*

To see this, we need to first realize that both the MC and TD value estimates are computed as running averages with the general form of Equation 3.

$$V(s_t) \leftarrow V(s_t) + \alpha[U(s_t) - V(s_t)] \tag{3}$$

Here, $U(s_t)$ is the value update associated with state s_t. In MC learning, the update is $U_{MC} = r_{s_t \to s_{t+1}} + \gamma G(s_{t+1})$. Meanwhile, in TD learning the update is $U_{TD} = r_{s_t \to s_{t+1}} + \gamma V_{TD}(s_{t+1})$. If we can show that the expected variance of each TD update is less than that of each MC update, then the same statement will hold for the running averages of the updates. So we'd like to prove that

$$\mathbb{E}[Var[U_{TD}]] \leq \mathbb{E}[Var[U_{MC}]] \tag{4}$$

Substituting in the definitions of U_{MC} and U_{TD} and canceling common terms, we can get Equation 5.

$$\mathbb{E}[Var[V_{TD}(s_{t+1})]] \leq \mathbb{E}[Var[G(s_{t+1})]] \tag{5}$$

Referring to Algorithm 1, note that $G(s')$ will have a fixed variance, which we will call σ^2_{Return}. So now we just have to show that

$$\mathbb{E}[Var[V_{TD}(s_{t+1})]] \leq \sigma^2_{Return} \tag{6}$$

Let's try analyzing $Var[V_{TD}(s_{t+1})]$ in terms of σ^2_{Return}. There are two general cases we need to consider. In one case, the MC and TD updates are identical; this turns Equation 6 into an equality. In the second case, the MC and TD updates look different, and Equation 6 is a strict inequality.

Case 1. The first thing to notice is that the TD and MC value updates are often identical. When this is true, their expected variances are the same. This happens whenever none of the states that occur after s_{t+1} along current trajectory T have been visited before. This also happens when, for some number of states after s_{t+1}, trajectory T overlaps perfectly with a previous trajectory or trajectories before diverging.

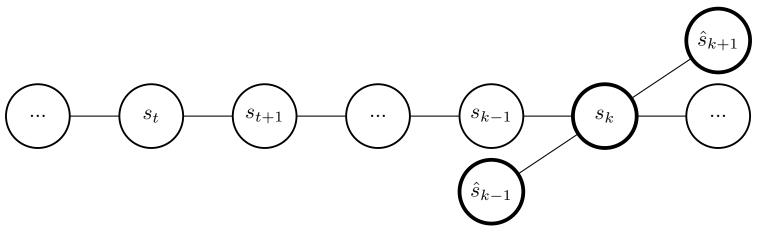

Figure 5: An illustration of Case 2: a previous trajectory T_{prev} (bolded nodes) merges with the current trajectory T at state s_k. During the value update associated with T_{prev}, $V_{TD}(s_k)$ gets updated. During the current value update, it gets updated again. Thus, when we set out to update $V_{TD}(s_{k-1})$, $U_{TD} = r_{s_{k-1} \to s_k} + \gamma V_{TD}(s_k)$ has lower variance than $U_{MC} = r_{s_{k-1} \to s_k} + \gamma G(s_k)$. This inequality then holds for all states in T leading back to s_{k-1}, including s_{t+1}.

Case 2. The only case where TD and MC updates to $V(s_{t+1})$ are not equivalent is when the current trajectory intersects with one or more previous trajectories after passing through state s_{t+1}. Let's consider this scenario in detail.

Suppose an intersection occurs at state s_k with $k > t+1$. When the agent performs the value update for s_k along the current trajectory, this update will be at least the second update to the running average $V_{TD}(s_k)$. Since the variance of a running average is inversely proportional to the total number of updates, we can say that the expected variance $\mathbb{E}[Var[V_{TD}(s_k)]] \leq \frac{\sigma^2_{Return}}{2}$. Since the variance of this quantity is less than that of $G(s_k)$, the expected TD update to s_{k-1} will be better than the expected MC update. To be concrete,

$$\mathbb{E}[Var[U_{TD}(s_{k-1})]] < \mathbb{E}[Var[U_{MC}(s_{k-1})]] \tag{7}$$

This inequality holds for all states that precede s_{k-1}, all of the way back to s_{t+1}. So for Case 2, the expected variance of $U_{TD}(s_{t+1})$ is lower than σ^2_{Return}.

Combining Cases 1 and 2, we can see that Equation 5 must be true. Since this is the case, our original statement must be true: the expected variance of a TD value estimate at state s_t along trajectory T must be less than or equal to that of a MC value estimate.

References

[1] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, 2 edition, 2018.