# Reinforcement Learning

## Chapter 3: Model-free RL

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Fall 2025

# Terminating Monte-Carlo: *Sample Inefficiency*

+ *But, isn't that as you said before sample inefficient?*
– Sure it is!

*We are loosing lots of states in each sample trajectory!*

*A better solution is to use the recursive property of return and do*

*bootstrapping*

*This is what we see next!*

# Testing *Genie in the Box*

Let's think again a bit science-fictional: *assume a genie can tell us the value $v_\pi(s)$ for each state $s$, We want to test this genie via a numerical algorithm* ☺

---

*Bellman tells us that at any state $s$, we should see*

$$v_\pi(s) = \mathbb{E}\{R_{t+1} | S_t = s\} + \gamma \mathbb{E}\{v_\pi(S_{t+1}) | S_t = s\}$$

*Monte-Carlo tells us further that after $K$ sample trajectories $S_0, A_0 \xrightarrow{R_1} S_1$ initiated at $S_0 = s$ and terminated after only one step, we have*

$$\mathbb{E}\{R_{t+1} | S_t = s\} \approx \frac{1}{K} \sum_{k=1}^{K} R_1[k]$$

$$\mathbb{E}\{v_\pi(S_{t+1}) | S_t = s\} \approx \frac{1}{K} \sum_{k=1}^{K} v_\pi(S_1[k])$$

# Testing *Genie in the Box*

*We could hence find an estimator of* $v_\pi(s)$ *as*

$$v_\pi(s) \approx \hat{v}_\pi(s) = \frac{1}{K}\sum_{k=1}^{K} R_1[k] + \gamma v_\pi(S_1[k])$$

*And, of course we could simply evaluate this estimator online as*

$$\hat{v}_\pi(s) \leftarrow \hat{v}_\pi(s) + \frac{1}{K}\left(R_1 + \gamma v_\pi(S_1) - \hat{v}_\pi(s)\right)$$

*if we need linear averaging or alternatively as*

$$\hat{v}_\pi(s) \leftarrow \hat{v}_\pi(s) + \alpha\left(R_1 + \gamma v_\pi(S_1) - \hat{v}_\pi(s)\right)$$

*if we think of more general weighted averaging*

# Computing Values via Bootstrapping: *Algorithm I*

*We can write our genie-testing algorithm as*

TD_verI($\pi, s$):

1: *Initiate estimator of value as* $\hat{v}_\pi(s) = 0$
2: *Ask genie* $v_\pi(\bar{s})$ *for all* $\bar{s}$ *that can be followed after* $s$
3: ***for*** *episode* $= 1 : K$ ***do***
4: *Initiate with state* $S_0 = s$ *and act via policy* $\pi(a|s)$
5: *Sample a single-step terminated trajectory*

$$S_0, A_0 \xrightarrow{R_1} S_1$$

6: *Update estimate of value as* $\hat{v}_\pi(s) \leftarrow \hat{v}_\pi(s) + \alpha(R_1 + v_\pi(S_1) - \hat{v}_\pi(s))$
7: ***end for***

*Note that in this algorithm we don't need the environment to be episodic, as we use recursive property of value-function!*

*This explains the idea of bootstrapping*

# Bootstrapping: *Using Value Estimates*

+ *But, in practice we don't have genie! And, say we find one, why should we compute values anymore?!*
- Absolutely! But, we may use this property

---

*We can replace those true values with their estimates*

- *They are initially bad estimates*
  ↳ *and thus return in bad estimate for the other state*
- *They gradually improve*
  ↳ *and therefore return better estimate for the other state*

*The key point is that we get rid of need for a terminal state!*

# Computing Values via Bootstrapping: *Algorithm II*

*So, we could get rid of the genie finally*

TD_verII($\pi, s$):

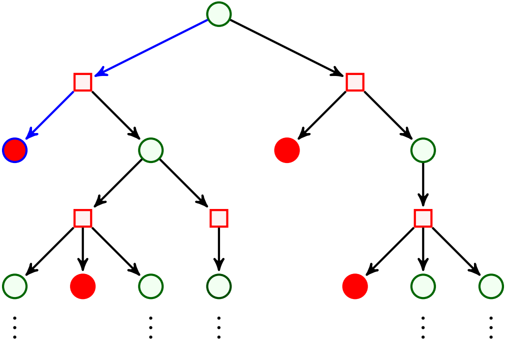
1: *Initiate estimator of value as* $\hat{v}_\pi(s) = 0$
2: *Use available* $\hat{v}_\pi(\bar{s})$ *for all* $\bar{s}$ *that follow* $s$
3: ***for*** *episode* $= 1 : K$ ***do***
4: *Initiate with state* $S_0 = s$ *and act via policy* $\pi(a|s)$
5: *Sample a single-step terminated trajectory*

$$S_0, A_0 \xrightarrow{R_1} S_1$$

6: *Update estimate of value as* $\hat{v}_\pi(s) \leftarrow \hat{v}_\pi(s) + \alpha(R_1 + \hat{v}_\pi(S_1) - \hat{v}_\pi(s))$
7: ***end for***

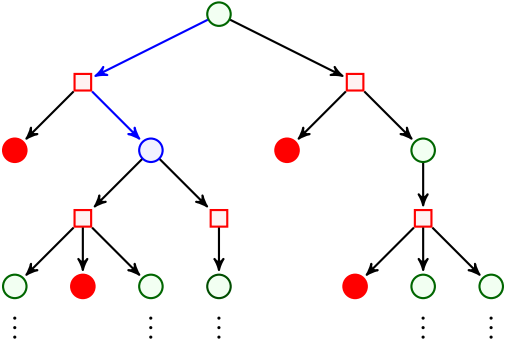
# Backup Diagram: *Sampling with Bootstrapping*

*Looking at the backup tree, we are now sampling one-step trajectories*



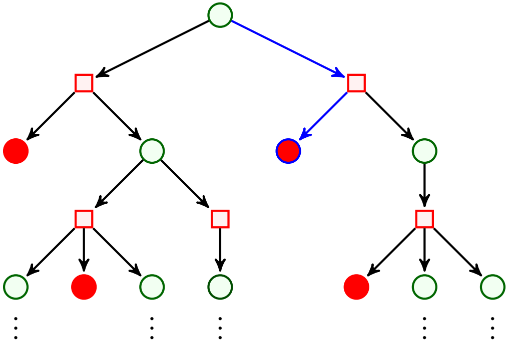
# Backup Diagram: *Sampling with Bootstrapping*

*Looking at the backup tree, we are now sampling one-step trajectories*



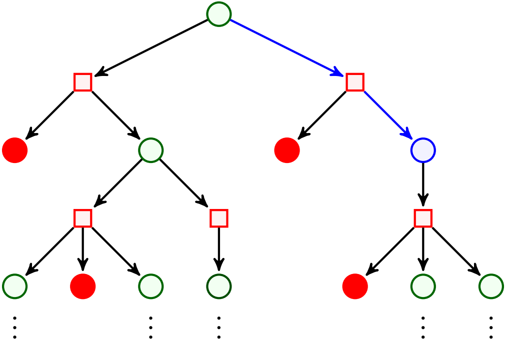
# Backup Diagram: *Sampling with Bootstrapping*

*Looking at the backup tree, we are now sampling one-step trajectories*



# Backup Diagram: *Sampling with Bootstrapping*

*Looking at the backup tree, we are now sampling one-step trajectories*



# Temporal Difference

Bootstrapping can replace Monte-Carlo in our evaluation algorithms

1. *We start with some initial value estimates*
2. *We sample a trajectory of finite length $T$*

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

↳ *it can either end with a terminal state if we have any*
↳ *or could simply be terminated*

3. *We move over trajectory and update value of each state by bootstrapping*

*This idea of estimating values is called*

*Temporal Difference* $\equiv$ *TD*

# Temporal Difference: *TD*-$0$

TD_Eval($\pi$):

*Initiate estimator of value as* $\hat{v}_\pi(s^n) = 0$ *for* $n = 1 : N$
***for*** *episode* $= 1 : K$ ***do***
    *Initiate with a random state* $S_0$ *and act via policy* $\pi(a|s)$
    *Sample a trajectory until either a terminal stated or some terminating* $T$

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

    ***for*** $t = 0 : T-1$ ***do***
        *Update as* $\hat{v}_\pi(S_t) \leftarrow \hat{v}_\pi(S_t) + \alpha(R_{t+1} + \gamma\hat{v}_\pi(S_{t+1}) - \hat{v}_\pi(S_t))$
    ***end for***
***end for***

## Attention

*With TD, we even don't need to wait till a trajectory is sampled!*

# Evaluating Action-Values via Bootstrapping

+ *What about the action-values? Can we bootstrap again?*
- Sure!

---

*Recall Bellman equation I of action-value function*

$$q_\pi(s, a) = \mathbb{E}\{R_{t+1} | S_t = s, A_t = a\} + \gamma \mathbb{E}\{v_\pi(S_{t+1}) | S_t = s, , A_t = a\}$$

*So, we can use sample trajectory to estimate action-values as well: at time $t$, we can update estimate of pair $(S_t, A_t)$ as*

$$\hat{q}_\pi(S_t, A_t) \leftarrow \hat{q}_\pi(S_t, A_t) + \alpha(R_{t+1} + \gamma \hat{v}_\pi(S_{t+1}) - \hat{q}_\pi(S_t, A_t))$$

# Temporal Difference: *Action-Value*

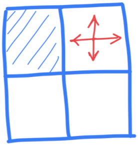
```
TD_QEval(π):
```

1: *Initiate estimator of value as* $\hat{q}_\pi(s^n, a^m) = 0$ *for* $n = 1 : N$ *and* $m = 1 : M$
2: ***for*** *episode* $= 1 : K$ ***do***
3: *Initiate with a random state-action pair* $(S_0, A_0)$ *and act via policy* $\pi(a|s)$
4: *Sample a trajectory until either a terminal stated or some terminating* $T$

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

5: ***for*** $t = 0 : T-1$ ***do***
6: *Set* $\hat{q}_\pi(S_t, A_t) \leftarrow \hat{q}_\pi(S_t, A_t) + \alpha(R_{t+1} + \gamma \hat{v}_\pi(S_{t+1}) - \hat{q}_\pi(S_t, A_t))$
7: ***end for***
8: ***end for***

# Example: *Dummy Grid World with Random Walk*



*Let's get back to our dummy world: we now use TD to compute the values for uniform random policy, i.e.,*

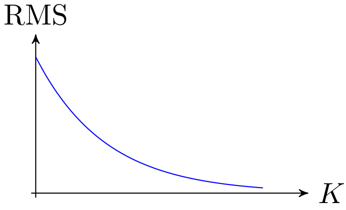
$$\pi(a|s) = \frac{1}{4}$$

*for all actions and states*

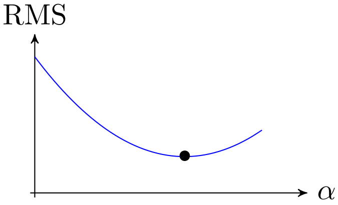
*Try it at home* ☺

# Typical Behavior

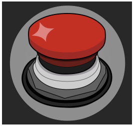
We are going to see the same behavior also with TD: *against $K$ we see*




*and against $\alpha$ we have a minimum*



# Example: *TD vs Monte-Carlo in Single-Button Game*



Consider the following dummy game: *we have got a single button to push; each time we push this button,*

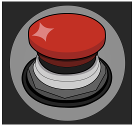
- *we either get into green or Blue mode allowing us to push the button again and returns a 0/1 reward*
- *or it gets into red mode which only returns a 0/1 reward and game is over*

---

*Obviously this game has*

- *Three states: green, Blue and red which is terminal*
- *a single action, i.e., pushing the button*

# Example: *TD vs Monte-Carlo in Single-Button Game*



*We play this game 6 episodes and get following sample trajectories*

$$\text{Blue} \xrightarrow{1} \text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Green} \xrightarrow{0} \text{Blue} \xrightarrow{0} \text{red}$$

*Let's estimate $v\left(\text{Blue}\right)$ and $v\left(\text{Green}\right)$ by both TD-$0$ and Monte-Carlo*

# Example: *TD vs Monte-Carlo in Single-Button Game*

Blue $\xrightarrow{1}$ Blue $\xrightarrow{0}$ red

Blue $\xrightarrow{1}$ red

Blue $\xrightarrow{0}$ Blue $\xrightarrow{1}$ red

Blue $\xrightarrow{1}$ red

Blue $\xrightarrow{0}$ red

Green $\xrightarrow{0}$ Blue $\xrightarrow{0}$ red

---

*With Monte-Carlo approach, we could say: we have 8 sample trajectories starting with* Blue *with 4 returning 1 and 4 returning 0; thus we have*

$$\hat{v}\left(\text{Blue}\right) = \frac{4}{8} = 0.5$$

*We also have only one sample trajectory starting at* Green *with zero return; thus, we have*

$$\hat{v}\left(\text{Green}\right) = \frac{0}{1} = 0$$

# Example: *TD vs Monte-Carlo in Single-Button Game*

$$\text{Blue} \xrightarrow{1} \text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Green} \xrightarrow{0} \text{Blue} \xrightarrow{0} \text{red}$$

---

*With TD-$0$, we could say: we have 8* Blue *states followed by either* Blue *or* red.
*If we bootstrap, we then get up to state* Blue *in the last trajectory*

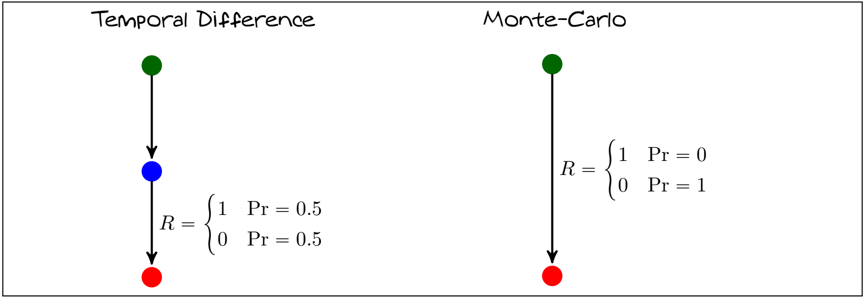
$$\hat{v}\left(\text{Blue}\right) \approx 0.5$$

*We then get to the last trajectory which is the only one with state* Green*: since we have not yet updated, we yet have $\hat{v}\left(\text{Green}\right) = 0$; by bootstrapping we get*

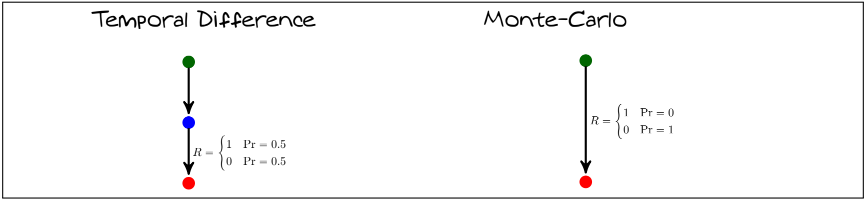
$$\hat{v}\left(\text{Green}\right) \leftarrow \underbrace{\hat{v}\left(\text{Green}\right)}_{0} + \left( \underbrace{R_{t+1}}_{0} + \underbrace{\hat{v}\left(S_{t+1}\right)}_{\hat{v}(\text{Blue})\approx 0.5} - \underbrace{\hat{v}\left(\text{Green}\right)}_{0} \right) \approx 0.5$$

# Temporal Difference vs Monte-Carlo: *Note I*

The observed difference follow a fundamental point: *in Monte-Carlo we only look at best approximation given data, without looking into the Markovity of the state, whereas in TD we take into account the fact that we are dealing with a Markov state*



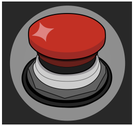
# Temporal Difference vs Monte-Carlo: *Note I*




*This is common to see in the literature that people say*

- *TD finds maximum-likelihood estimate of the values*
  - ↳ *It uses this assumption that the state is a Markov process*
  - ↳ *It's the better option, if we are sure that we have access to the complete (Markov) state*
- *Monte-Carlo finds least-squares estimate of the values*
  - ↳ *It ignores Markovity of the state*
  - ↳ *Maybe better option, when we cannot access the complete (Markov) state*

# Back to Single-Button Game: *Side Note on Batch Updating*



*What would happen, if we get sample trajectories in the following order*

$$\text{Green} \xrightarrow{0} \text{Blue} \xrightarrow{0} \text{red}$$

$$\text{Blue} \xrightarrow{1} \text{Blue} \xrightarrow{0} \text{red}$$

$$\text{Blue} \xrightarrow{1} \text{red}$$

$$\text{Blue} \xrightarrow{0} \text{Blue} \xrightarrow{1} \text{red}$$

$$\text{Blue} \xrightarrow{1} \text{red}$$

$$\text{Blue} \xrightarrow{0} \text{red}$$

# Back to Single-Button Game: *Side Note on Batch Updating*

$$\text{Green} \xrightarrow{0} \text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{red}$$

---

*If we go only once over the batch of all episodes with TD, we get*

$$\hat{v}\left(\text{Green}\right) \approx 0$$
$$\hat{v}\left(\text{Blue}\right) \approx 0.5$$

# Back to Single-Button Game: *Side Note on Batch Updating*

$$\text{Green} \xrightarrow{0} \text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{Blue} \xrightarrow{0} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{1} \text{red}$$
$$\text{Blue} \xrightarrow{0} \text{red}$$

---

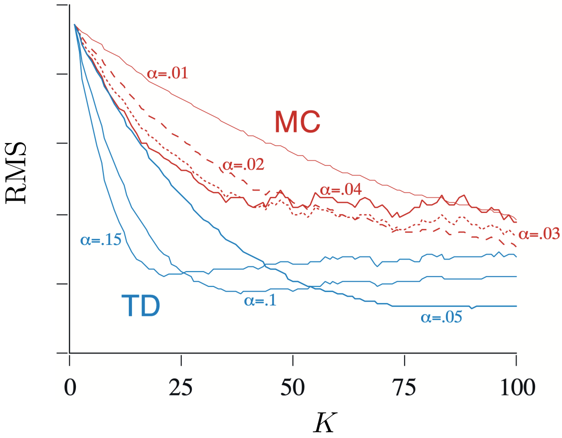
*If we go twice over the batch of all episodes with TD, we get*

$$\hat{v}\left(\text{Green}\right) \approx 0.5$$
$$\hat{v}\left(\text{Blue}\right) \approx 0.5$$

*We get better if we go over the batch od data multiple times!*

# Temporal Difference vs Monte-Carlo: *Note II*

*Let's see how Monte-Carlo performs against TD-$0$ algorithm for a bit larger example of random walk on a grid*[1]




[1] *This figure is taken from Chapter 6 of Sutton and Barto's book*

# Recall: *Bias and Variance of Estimator*

*At this point, we need to have some clue about bias and variance of an estimator*

*If you need to recap, please look at the board*

# Temporal Difference vs Monte-Carlo: *Note II*

What has been seen in the diagram is a general behavior

- *Monte-Carlo is good in sense of bias but bad in terms of variance*
  - ↳ *It always returns an unbiased estimator of value, i.e.,*

$$\mathbb{E}\left\{\hat{v}_\pi\left(s\right)\right\} = v_\pi\left(s\right)$$

  - ↳ *It's estimation is however high variance, i.e.,*

$$\mathbb{E}\left\{\left(\hat{v}_\pi\left(s\right) - v_\pi\left(s\right)\right)^2\right\} \rightsquigarrow \text{large}$$

- *TD-$0$ is good in sense of variance but can be bad in terms of bias*
  - ↳ *It can return a biased estimator of value, i.e.,*

$$\mathbb{E}\left\{\hat{v}_\pi\left(s\right)\right\} \neq v_\pi\left(s\right)$$

  - ↳ *It's estimation is low variance, i.e.,*

$$\mathbb{E}\left\{\left(\hat{v}_\pi\left(s\right) - v_\pi\left(s\right)\right)^2\right\} \rightsquigarrow \text{small}$$

# Policy Iteration with TD-$0$

*We can use TD-$0$ to implement another variant of GPI*

```
MC_PolicyItr():
1: Initiate two random policies π and π̄
2: while π ≠ π̄ do
3:     q̂_π = TD_QEval(π) and π ← π̄
4:     π̄ = Greedy(q̂_π)
5: end while
```

*And, recall that our greedy algorithm is*

```
Greedy(q_π):
1: for n = 1 : N do
2:     Improve the by taking deterministically the best action
```

$$\bar{\pi}\left(a^m | s^n\right) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_\pi\left(s^n, a^m\right) \\ 0 & m \neq \operatorname*{argmax}_m q_\pi\left(s^n, a^m\right) \end{cases}$$

```
3: end for
```

# GPI with TD-$0$: Visualization

*We can plot the same figure again in this case*