Model-Free
Tabular Method Solutions

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Fall 2019

avereshc@buffalo.edu

September 17, 2019

Overview

1. Recap

2. SARSA

Table of Contents

Recap: Monte Carlo (MC) and Temporal Difference (TD) Learning

- Goal: learn $v_\pi(s)$ from episodes of experience under policy π
- Incremental every-visit Monte-Carlo:
  - Update value $V(S_t)$ toward actual return G_t

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

- Simplest Temporal-Difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward estimated returns $R_{t+1} + \gamma V(S_{t+1})$

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

- $R_{t+1} + \gamma V(S_{t+1})$ is called the TD target
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error.

Recap: DP vs. MC vs TD Learning

- Remember:

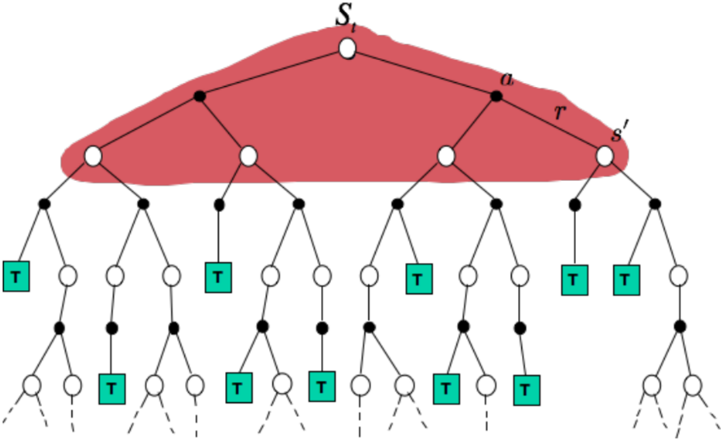
MC: sample average return approximates expectation

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s\right] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] .
\end{aligned}
$$

TD: combine both: Sample expected values and use a current estimate $V(S_{t+1})$ of the true $v_\pi(S_{t+1})$

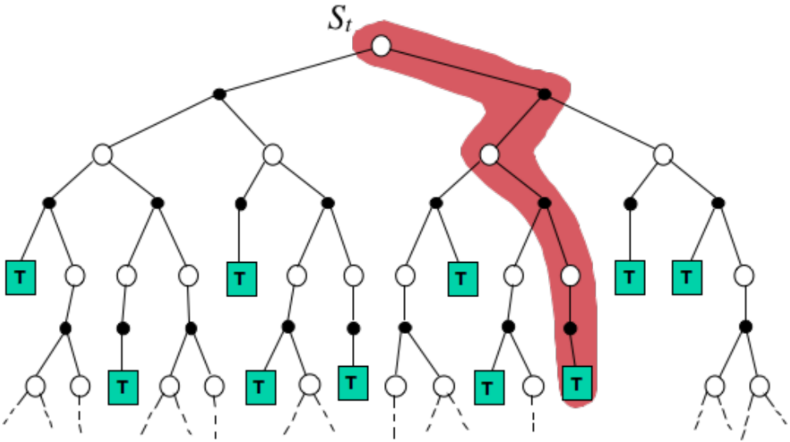
DP: the expected values are provided by a model. But we use a current estimate $V(S_{t+1})$ of the true $v_\pi(S_{t+1})$

$$V(S_t) \leftarrow E_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right] = \sum_a \pi(a|S_t) \sum_{s',r} p(s',r|S_t,a)[r + \gamma V(s')]$$



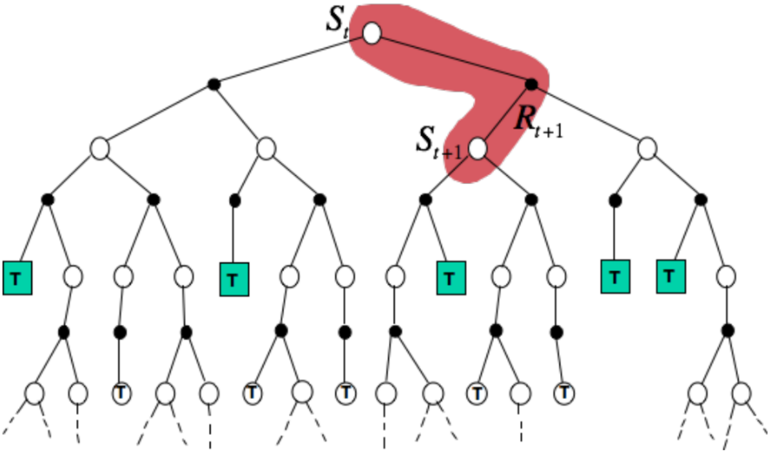
Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$



TD(0) Method

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



TD Methods Bootstrap and Sample

- **Bootstrapping:** update involves an estimate
  - MC does not bootstrap
  - DP bootstrap
  - TD bootstrap

TD Methods Bootstrap and Sample

- **Bootstrapping:** update involves an estimate
  - MC does not bootstrap
  - DP bootstrap
  - TD bootstrap
- **Sampling:** update does not involve an expected value
  - MC samples
  - DP does not sample
  - TD samples

TD(0) Method

- Policy Evaluation (the prediction problem):
  - for a given policy π, compute the state-value function v_π

- Remember: Simple every-visit Monte Carlo method:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$

target: the actual return after time *t*

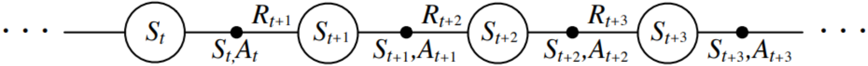
- The simplest Temporal-Difference method TD(0):

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\textbf{target}\text{: an estimate of the return}} - V(S_t)\Big]$$

Table of Contents

Learning an Action-Value Function

- Estimate q_π for the current policy π



After every transition from a nonterminal state, S_t, do this:

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

If S_{t+1} is terminal, then define $Q(S_{t+1}, A_{t+1}) = 0$

SARSA

Turn this into a control method by always updating the policy to be **greedy** with respect to the current estimate

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

SARSA

- Instead of the sample value-of-next-state, use the expectation!

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big] \\
&\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big]
\end{aligned}
$$

- Expected Sarsa performs better than Sarsa (but costs more)

SARSA

- SARSA is an **on-policy** algorithm which means that while learning the optimal policy it uses the current estimate of the optimal policy to generate the behaviour
- SARSA converges to an **optimal policy** as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the greedy policy ($\epsilon = \frac{1}{t}$).