Robot Perception and Learning

Policy Iteration, Monte Carlo Methods and Temporal Difference Learning

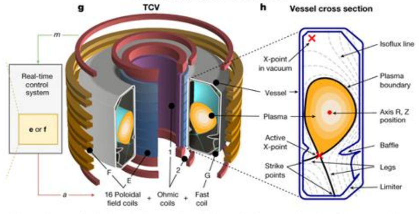
Tsung-Wei Ke

Fall 2025

Recap

RL as a general learning framework for different tasks

Power Plant Control



Magnetic control of tokamak plasmas through deep reinforcement learning. Degrave et al.

Trading

Gaming

Adaptive VisuoMotor Control

AnyCar to Anywhere: Learning Universal Dynamics Model for Agile and Adaptive Mobility. Xiao et al.

Content Generation

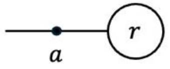
Training Animals

The learning objective of RL

$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{p_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Multi-armed Bandit Problem

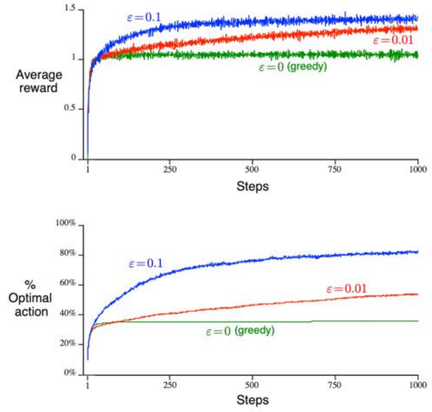


- Expected reward: $q^*(a_k) = \mathbb{E}[r_t | A_t = a_k]$
- Action-value estimates: $Q_t(a_k)$
- **Greedy action selection method**: select the action with the highest estimated value:

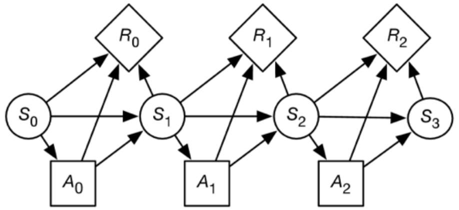
$$A_t^* = \arg\max_a Q_t(a)$$

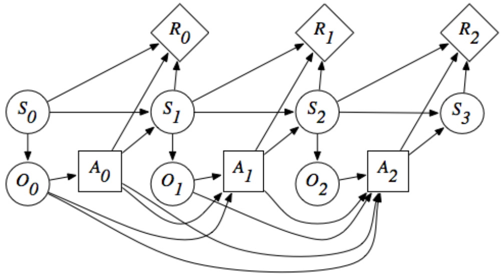
➢ If $A_t = A_t^*$, you are *exploiting* your current knowledge of the values of the actions

➢ If $A_t \neq A_t^*$, you are *exploring*. You improve your estimate of the non-greedy actions

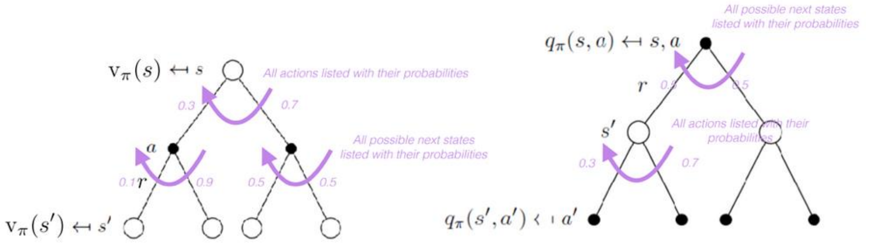


Markov Decision Process





- Discounted returns: $G_t = R_{t+1} + \gamma G_{t+1}$
- The state value function $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$



$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a) \left[r + \gamma v_\pi(s') \right]$$

$$q_\pi(s,a) = \sum_{r,s'} p(s', r | s, a) \left(r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a') \right)$$

Last Time, We Said the Greedy Strategy Improves the Current Policy

- Let's say we obtain the value function $v_\pi(s)$ based on policy π using dynamic programming, How can we improve the policy?

- Switch to a greedy policy!

$$\pi'(a|s) = \begin{cases} 1, if\ a = \underset{a}{\operatorname{argmax}}(\Sigma_{s',r} p(s',r|s,a))(r + \gamma v_\pi(s')) \\ 0, otherwise. \end{cases}$$

- Why greedy policy π' is better than the original policy π at state s?

  Since a greedy policy is deterministic: $\pi'(s) = \underset{a}{\operatorname{argmax}}(\Sigma_{s',r} p(s',r|s,a))(r + \gamma v_\pi(s'))$

$$\boxed{q_\pi(s|\pi'(s))} = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

$$\geq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] = \boxed{v_\pi(s)}$$

The value of selecting action $\pi'(s)$ is higher than following policy π at state s (here we still follow policy π at other states)

In fact, the State Value Function following the Greedy Policy $\boldsymbol{\pi'}$ is Better than Original Policy $\boldsymbol{\pi}$

$$
\begin{aligned}
v_\pi(s) &\le q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]
\end{aligned}
$$

$$
\pi'(a|s) = \begin{cases} 1, if\ a = \underset{a}{\operatorname{argmax}}(\Sigma_{s',r} p(s', r|s, a))(r + \gamma v_\pi(s')) \\ 0, otherwise. \end{cases}
$$

In fact, the State Value Function following the Greedy Policy π' is Better than Original Policy π

$$
\begin{aligned}
v_\pi(s) &\le q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\le \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s]
\end{aligned}
$$

In fact, the State Value Function following the Greedy Policy π' is Better than Original Policy π

$$
\begin{aligned}
v_\pi(s) &\le q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\le \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s\big]
\end{aligned}
$$

In fact, the State Value Function following the Greedy Policy π' is Better than Original Policy π

$$
\begin{aligned}
v_\pi(s) &\le q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\le \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \bigm| S_t = s\big] \\
&\le \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \bigm| S_t = s\big]
\end{aligned}
$$

In fact, the State Value Function following the Greedy Policy π' is Better than Original Policy π

$$
\begin{aligned}
v_\pi(s) &\le q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\le \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \bigm| S_t = s\big] \\
&\le \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \bigm| S_t = s\big] \\
&\vdots \\
&\le \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \bigm| S_t = s\big] \\
&= v_{\pi'}(s).
\end{aligned}
$$

Policy Evaluation and Policy Improvement

- **Policy Evaluation**: update the (state) value function following the current policy π

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

- **Policy Improvement**: improve the current policy π by acting greedily

$$\pi'(a|s) = \begin{cases} 1, if\ a = \underset{a}{\operatorname{argmax}}(\Sigma_{s',r} p(s',r|s,a))(r + \gamma v_\pi(s')) \\ 0, otherwise. \end{cases}$$

- What if the new greedy policy π' is no better than the original policy π ($v_{\pi'} = v_\pi$)?

$$v_{\pi'}(s) = \sum_a \pi'(a|s) \sum p(s',r|s,a)[r + \gamma v_{\pi'}(s')]$$

$$= \max_a(\sum_{s',r} p(s',r|s,a))(r + \gamma v_{\pi'}(s'))$$

We have an optimal policy!

Remember Bellman Optimality Equation for v^*

1. We have $v^*(s) = \max_a q_{\pi^*}(s,a)$

Why?

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

$$\Rightarrow v^*(s) = \sum_{a \in \mathcal{A}} \pi^*(a|s) q_{\pi^*}(s,a) = \max_a q_{\pi^*}(s,a)$$

$$\pi^*(a|s) = \begin{cases} 1, if\ a = \operatorname{argmax}_a q^*(s,a). \\ 0, otherwise. \end{cases}$$

Can We Approach Optimality by Alternating Policy Evaluation and Improvement?

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

- Policy Evaluation: update the (state) value function following the current policy π
- Policy Improvement: improve the current policy π by acting greedily

Policy Iteration

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

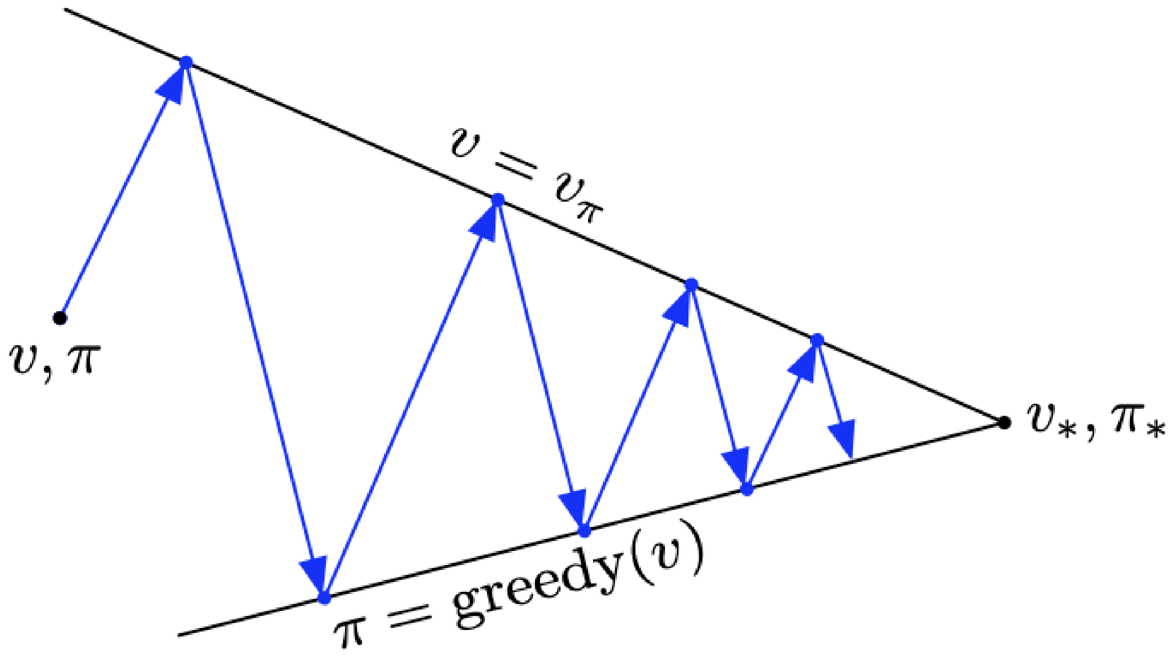
2. Policy Evaluation
   Loop:
   - $\Delta \leftarrow 0$
   - Loop for each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

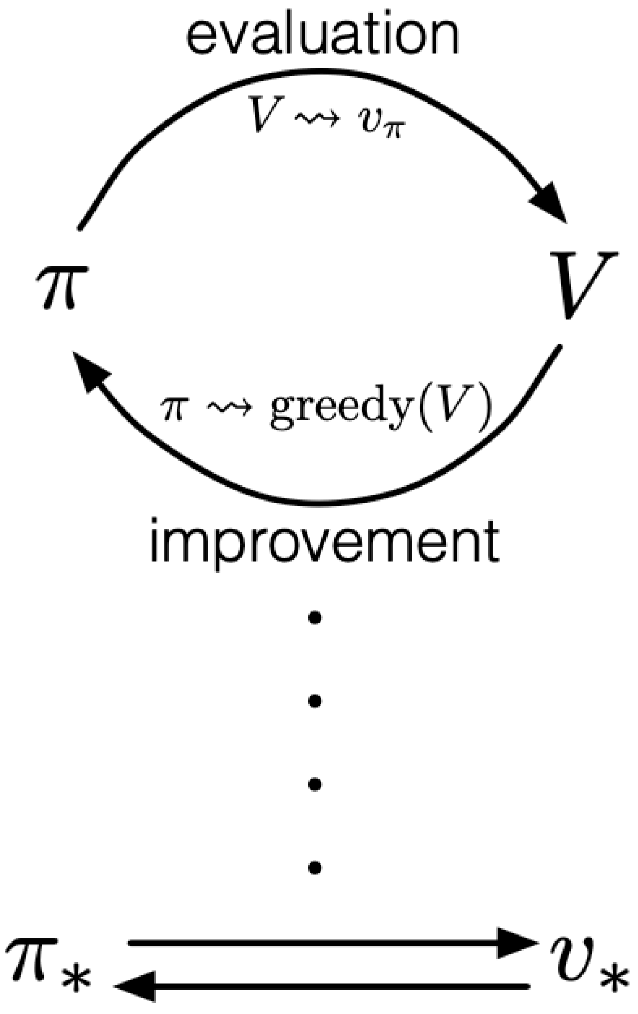
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Generalized Policy Iteration





- **Generalized Policy Iteration:** general idea of letting policy-evaluation and policy-improvement processes interact, independent of the granularity and other details of the two processes

Value Iteration

Things still work out even if we are lazy and partially complete policy iteration steps

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

Proof of Optimality?

Definition 1. *A Bellman optimality operator* $\mathcal{T} : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ *is an operator that satisfies: for any* $V \in \mathbb{R}^{|S|}$,

$$(\mathcal{T}V)(s) = \max_a \left[r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} V(s') \right].$$

Value iteration can thus be represented as recursively applying the Bellman optimality operator:

$$V_{k+1} = \mathcal{T} V_k. \tag{3}$$

The Bellman optimality operator $\mathcal{T}$ has several excellent properties. It is easy to verify that V^* is a fixed point of $\mathcal{T}$, i.e., $\mathcal{T}V^* = V^*$. Another important property is that $\mathcal{T}$ is a contraction mapping.

IE 498 Tanmay Gangwani:
https://yuanz.web.illinois.edu/teaching/IE498fa19/lec_16.pdf

Proof of Optimality?

In finite dimensional coordinate space, let $x = (x_1, \cdots, x_n)$:
$\|x\|_\infty := \max(|x_1|, \cdots, |x_n|)$

Theorem 2. *$\mathcal{T}$ is a contraction mapping under sup-norm $\|\cdot\|_\infty$, i.e., there exists $\gamma \in [0, 1)$ such that*

$$\|\mathcal{T}U - \mathcal{T}V\|_\infty \le \gamma \|U - V\|_\infty, \forall U, V \in \mathbb{R}^{|S|}.$$

Theorem 4. *Value iteration (3) converges to V^*, i.e.,*

$$\lim_{k\to\infty} V_k = V^*,$$

where $V_k = \mathcal{T}^{k-1}V_0$.

Our goal is to show

Proof of Optimality?

Theorem 4. *Value iteration (3) converges to V^*, i.e.,*

$$\lim_{k\to\infty} V_k = V^*,$$

where $V_k = \mathcal{T}^{k-1}V_0$.

Proof. Note that V^* is a fixed point of $\mathcal{T}$. In addition, according to Theorem 2, $\mathcal{T}$ is a contraction mapping. Therefore,

$$\|V_k - V^*\|_\infty = \|\mathcal{T}V_{k-1} - \mathcal{T}V^*\|_\infty \le \gamma\|V_{k-1} - V^*\|_\infty \le \cdots \le \gamma^k\|V_0 - V^*\|_\infty.$$

Let $k \to \infty$, and we have $\|V_k - V^*\|_\infty \to 0$. Thus $\lim_{k\to\infty} V_k = V^*$. □

IE 498 Tanmay Gangwani: https://yuanz.web.illinois.edu/teaching/IE498fa19/lec_16.pdf

We Need to Prove $\mathcal{T}$ is a Contraction Mapping

Theorem 2. *$\mathcal{T}$ is a contraction mapping under sup-norm $\|\cdot\|_\infty$, i.e., there exists $\gamma \in [0,1)$ such that*

$$\|\mathcal{T}U - \mathcal{T}V\|_\infty \leq \gamma \|U - V\|_\infty, \forall U, V \in \mathbb{R}^{|S|}.$$

Proof. To prove this property, we need the following lemma:

Lemma 3.

$$\left|\max_a f(a) - \max_a g(a)\right| \leq \max_a |f(a) - g(a)|.$$

Lemma 3.

$$\left|\max_a f(a) - \max_a g(a)\right| \leq \max_a |f(a) - g(a)|\,.$$

- Assume without loss of generality $\max_a f(a) \geq \max_a g(a)$ and denote $a^* = \operatorname{argmax}_a f(a)$

$$\left|\max_a f(a) - \max_a g(a)\right| = \max_a f(a) - \max_a g(a) = f(a^*) - \max_a g(a) \leq f(a^*) - g(a^*) \leq \max_a |f(a) - g(a)|\,.$$

Lemma 3.

$$\left|\max_a f(a) - \max_a g(a)\right| \leq \max_a |f(a) - g(a)|.$$

- Assume without loss of generality $\max_a f(a) \geq \max_a g(a)$ and denote $a^* = \operatorname{argmax}_a f(a)$

$$\left|\max_a f(a) - \max_a g(a)\right| = \max_a f(a) - \max_a g(a) = f(a^*) - \max_a g(a) \leq f(a^*) - g(a^*) \leq \max_a |f(a) - g(a)|.$$

- Let's prove theorem 2

$$\begin{aligned}
|\mathcal{T}V(s) - \mathcal{T}U(s)| &= \left|\max_a \left[r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} V(s')\right] - \max_a \left[r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} U(s')\right]\right| \\
&\leq \max_a \left|\gamma \mathbb{E}_{s' \sim T(s'|s,a)} \left[V(s') - U(s')\right]\right| \\
&\triangleq \left|\gamma \mathbb{E}_{s' \sim T(s'|s,a^*)} \left[V(s') - U(s')\right]\right| \quad \text{where, } a^* \text{ is the argmax of the RHS above} \\
&\leq \gamma \max_{s'} |V(s') - U(s')| \\
&= \gamma \|V - U\|_\infty
\end{aligned}$$

Lemma 3.

$$\left|\max_a f(a) - \max_a g(a)\right| \leq \max_a |f(a) - g(a)|\,.$$

- Let's prove theorem 2

$$\boxed{|\mathcal{T}V(s) - \mathcal{T}U(s)|} = \left|\max_a \left[r(s,a) + \gamma\mathbb{E}_{s'\sim T(s'|s,a)}V(s')\right] - \max_a \left[r(s,a) + \gamma\mathbb{E}_{s'\sim T(s'|s,a)}U(s')\right]\right|$$

Hold for any state s!

$$\leq \max_a \left|\gamma\mathbb{E}_{s'\sim T(s'|s,a)}\left[V(s') - U(s')\right]\right|$$

$$\triangleq \left|\gamma\mathbb{E}_{s'\sim T(s'|s,a^*)}\left[V(s') - U(s')\right]\right| \quad \text{where, } a^* \text{ is the argmax of the RHS above}$$

$$\leq \gamma \max_{s'} |V(s') - U(s')|$$

$$= \gamma\|V - U\|_\infty$$

- Lemma 3 also holds when:

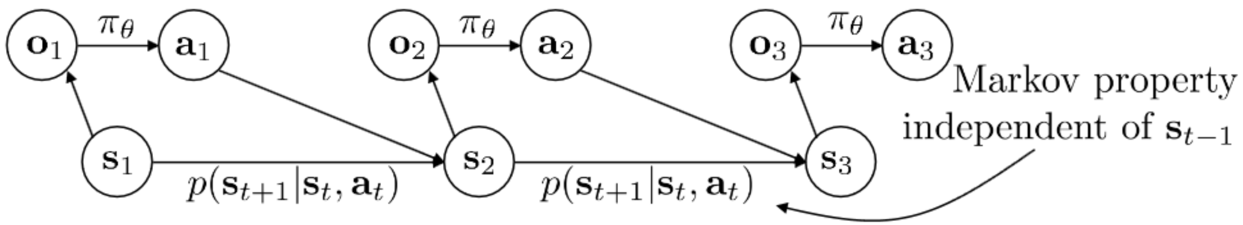
$$\max_s |\mathcal{T}V(s) - \mathcal{T}U(s)| \leq \gamma\|V - U\|_\infty, \longrightarrow \|\mathcal{T}U - \mathcal{T}V\|_\infty \leq \gamma\|V - U\|_\infty.$$

IE 498 Tanmay Gangwani: https://yuanz.web.illinois.edu/teaching/IE498fa19/lec_16.pdf

So far, we have several assumptions

We can solve MDP if we know:

1. The transition function (dynamics) of the environment
2. The reward function
3. The Markov property holds
4. We have enough computational resource



Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

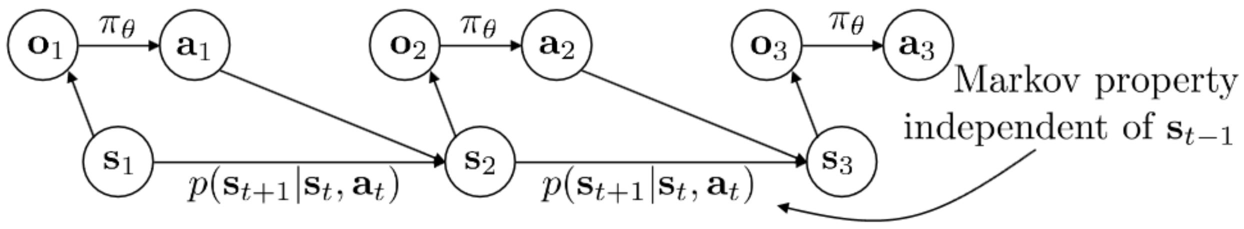
Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
 $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

Next, we'll go beyond these assumptions

We can solve MDP if we know:

1. The transition function (dynamics) of the environment
2. The reward function
3. The Markov property holds
4. We have enough computational resource

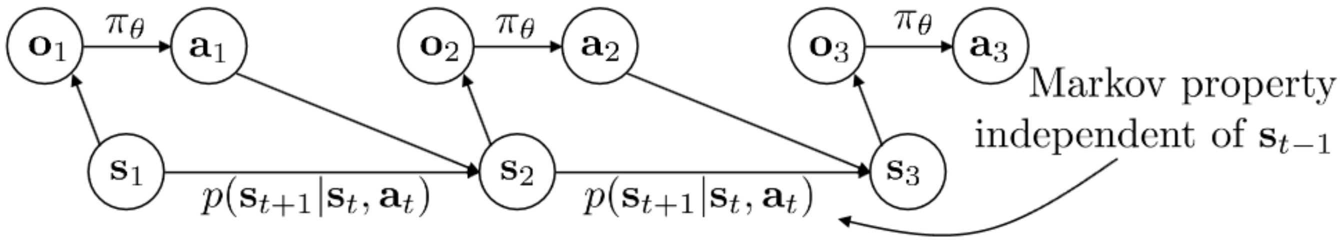


Monte Carlo Methods!

Temporal-Difference Learning!

What if we don't know the transition function and reward function?

Markov Decision Processes:



Bellman Optimality Equation

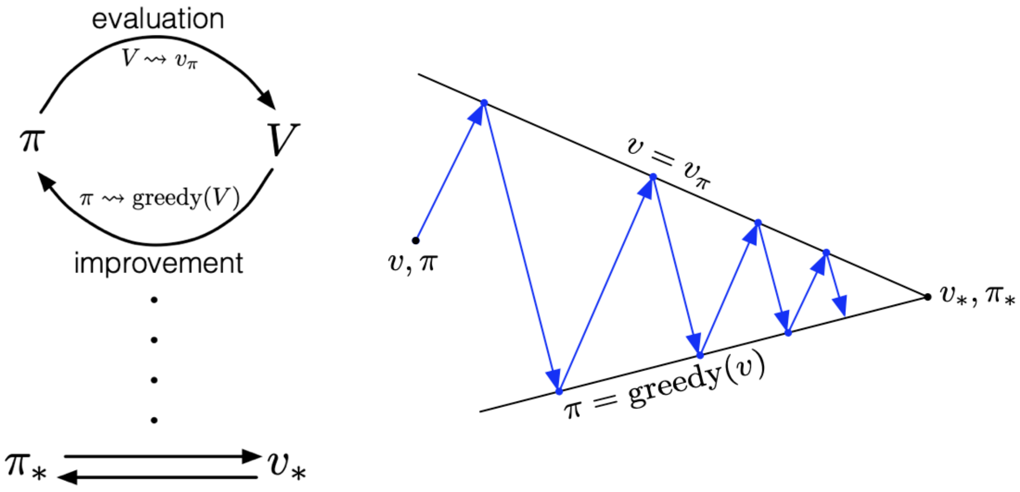
$$v_*(s) = \max_a q_{\pi_*}(s,a)$$
$$= \max_a \sum_{s',r} \cancel{p(s',r|s,a)}[\cancel{r(s,a)} + \gamma v_*(s')]$$

$$q_*(s,a) = \sum_{s',r} \cancel{p(s',r|s,a)}[\cancel{r(s,a)} + \gamma \max_{a'} q_*(s',a')]$$

Assumptions:

Full observation $o_t = s_t$
~~Known transition function $p(s_{t+1}|s_t, a_t)$~~
~~Known reward function $r(s_t, a_t)$~~

Generalized Policy Iteration



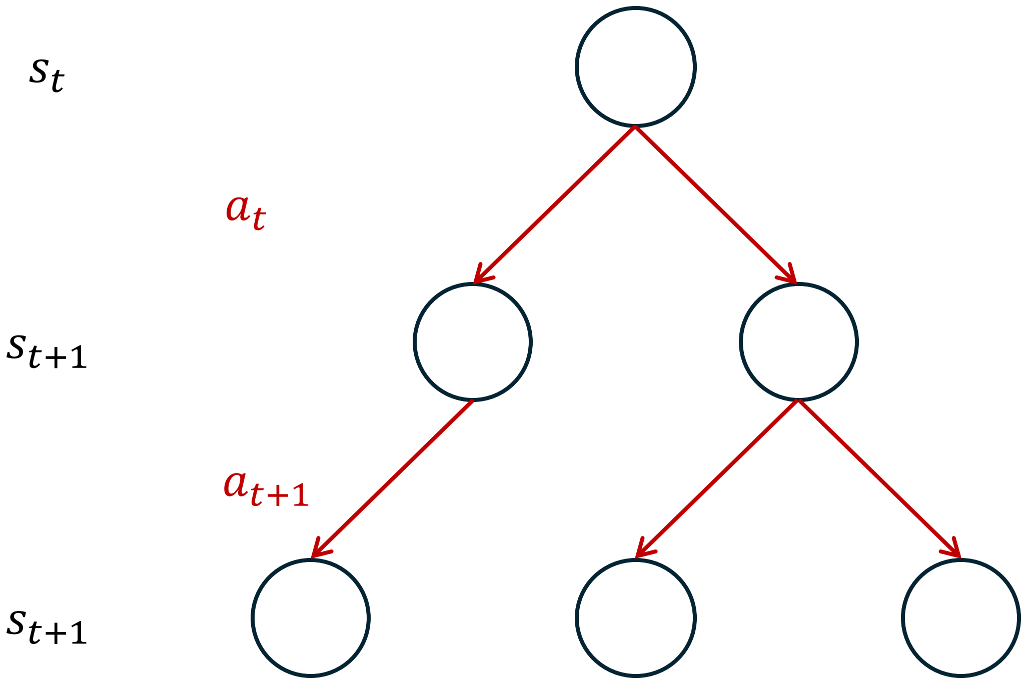
Value Functions

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} \cancel{p(s',r|s,a)}[\cancel{r(s,a)} + \gamma v_\pi(s')]$$

$$q_\pi(s,a) = \sum_{s',r} \cancel{p(s',r|s,a)}[\cancel{r(s,a)} + \gamma v_\pi(s')]$$

Monte Carlo (MC) Methods: Learning from experience

Dynamic Programming Method

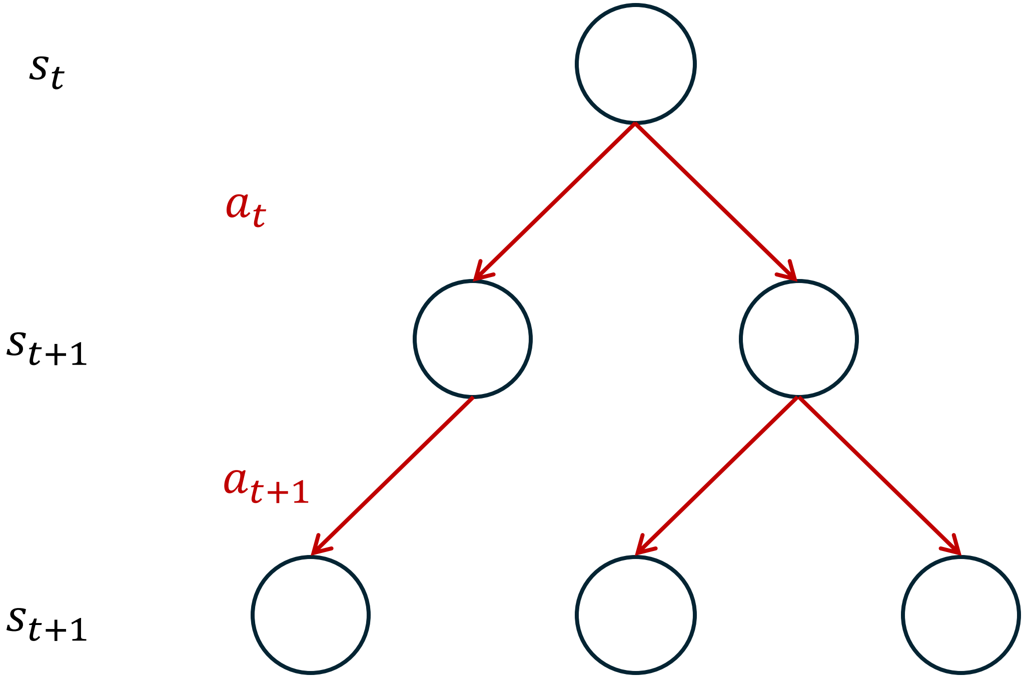


Simulated interaction: we know s_{t+1} as we know $p(s_{t+1}|s_t, a_t)$

- No (need for) exploration
- No (need for) interaction

Monte Carlo (MC) Methods: Learning from experience

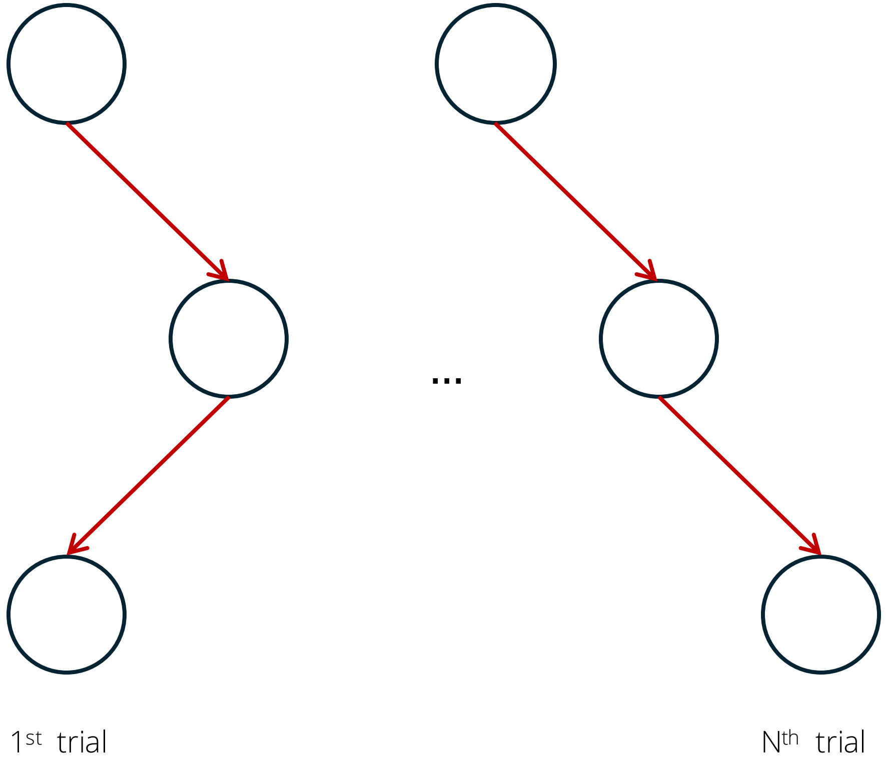
<u>Dynamic Programming Method</u>



Simulated interaction: we know s_{t+1} as we know $p(s_{t+1}|s_t, a_t)$

- No (need for) exploration
- No (need for) interaction

<u>Monte Carlo Method</u>



Actual experience: we don't know s_{t+1}. unless we visit it

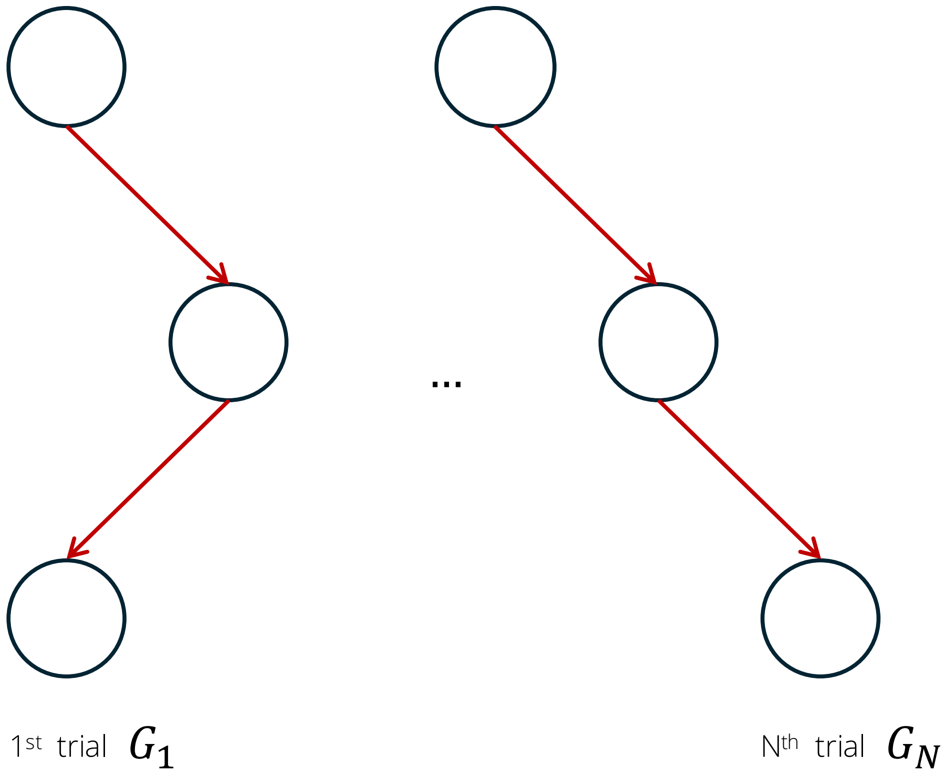
- Need exploration and interaction

Monte Carlo (MC) Methods: Learning from experience

- MC is model-free: no knowledge of MDP transitions / rewards
- MC learns from complete episodes: no bootstrapping (the estimates for each state is independent)

- What is "*bootstrapping*"?
  - ➢ Update value estimates on the basis of other estimates

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$



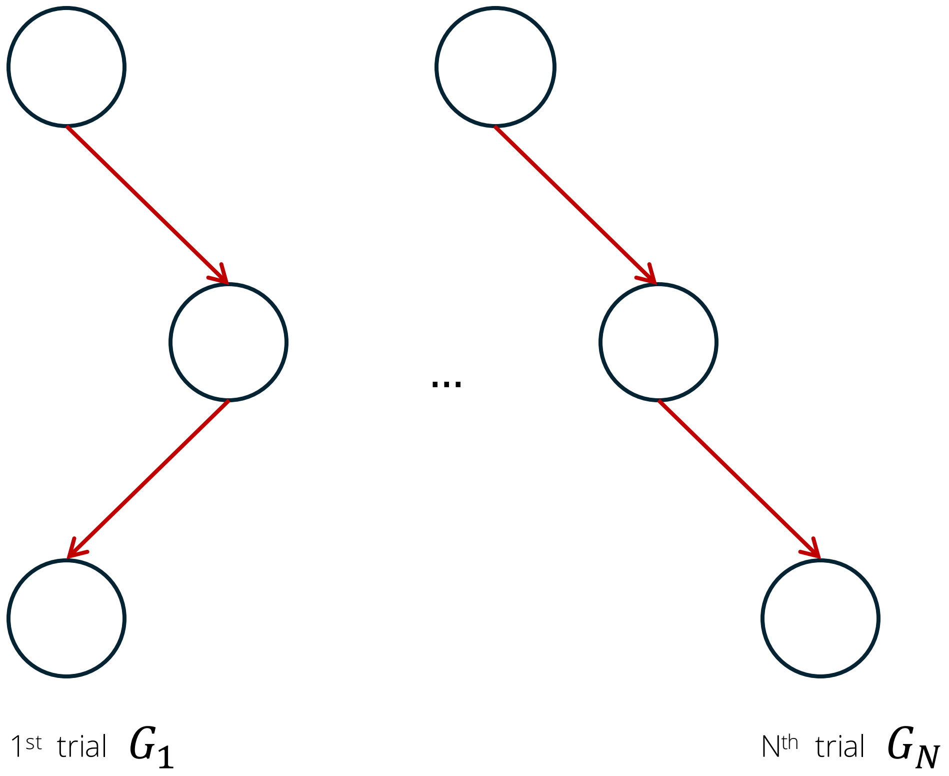
Monte Carlo (MC) Methods: Learning from experience

- How to estimate values?
  - The same idea of sample-average return:

$$V_N(s) = \frac{G_1 + \ldots + G_N}{N}$$

average observed returns from state s

  - The estimate converges to the true value with enough number of samples
    $V_N(s) \rightarrow v_\pi(s) \quad as \quad N \rightarrow \infty$



Monte Carlo (MC) Methods: Learning from experience

- An incremental implementation:

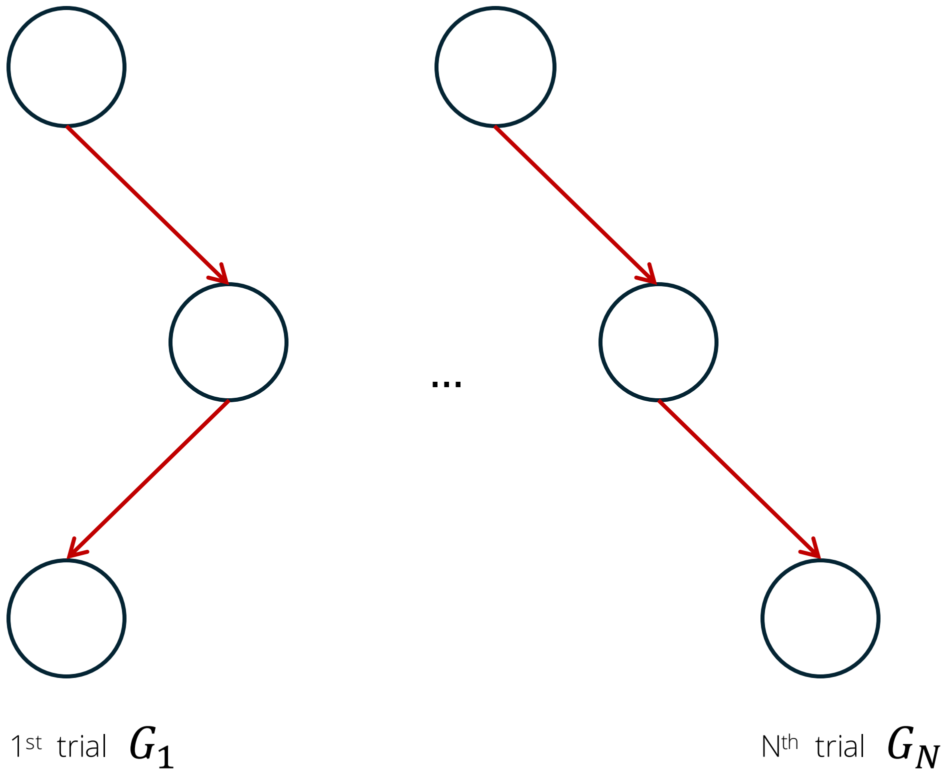
$$
\begin{aligned}
V_N(s) &= \frac{G_1 + \ldots + G_N}{N} \\
&= \frac{N-1}{N}\frac{G_1 + \ldots + G_{N-1}}{N-1} + \frac{1}{N} G_N \\
&= \frac{N-1}{N} V_{N-1}(s) + \frac{1}{N} G_N \\
&= V_{N-1}(s) + \frac{1}{N}(G_N - V_{N-1}(s))
\end{aligned}
$$

- A more general form:

$$NewEstimate \leftarrow OldEstimate + StepSize \left[Target - OldEstimate \right].$$

Set $StepSize < 1$ to forget old estimations.
Useful for non-stationary problems!

Non-stationary problems: $r(s,a)$ or $p(s'|s,a)$ changes over time



Backup diagram for Monte Carlo methods

- The entire trajectory of an episode is included
- Only applies to episodic MDPs
(all episodes must terminate)
- Only sampled transitions are included
- Does not bootstrap from successor state's value.
(Estimates for each state are independent)

MC Backup

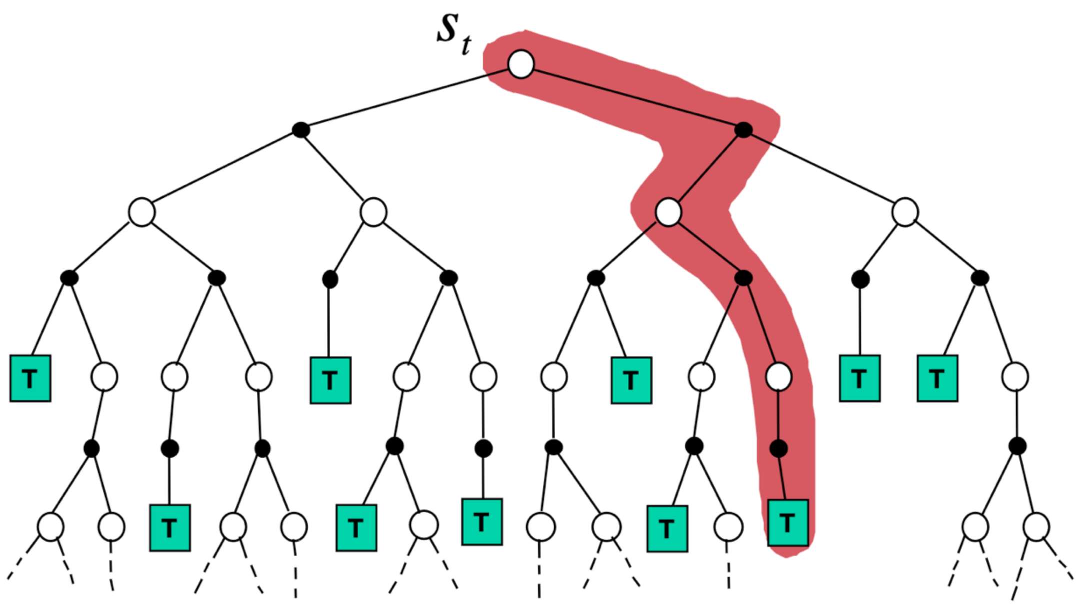
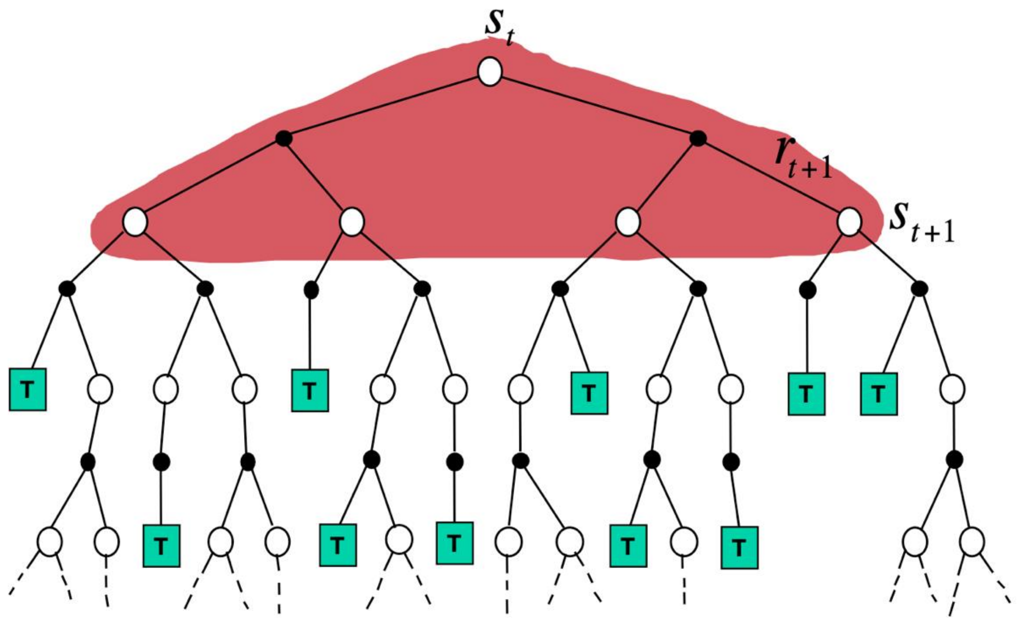


Image credit D. Silver

Backup diagram: DP vs. MC

Dynamic Programming Backup



MC Backup

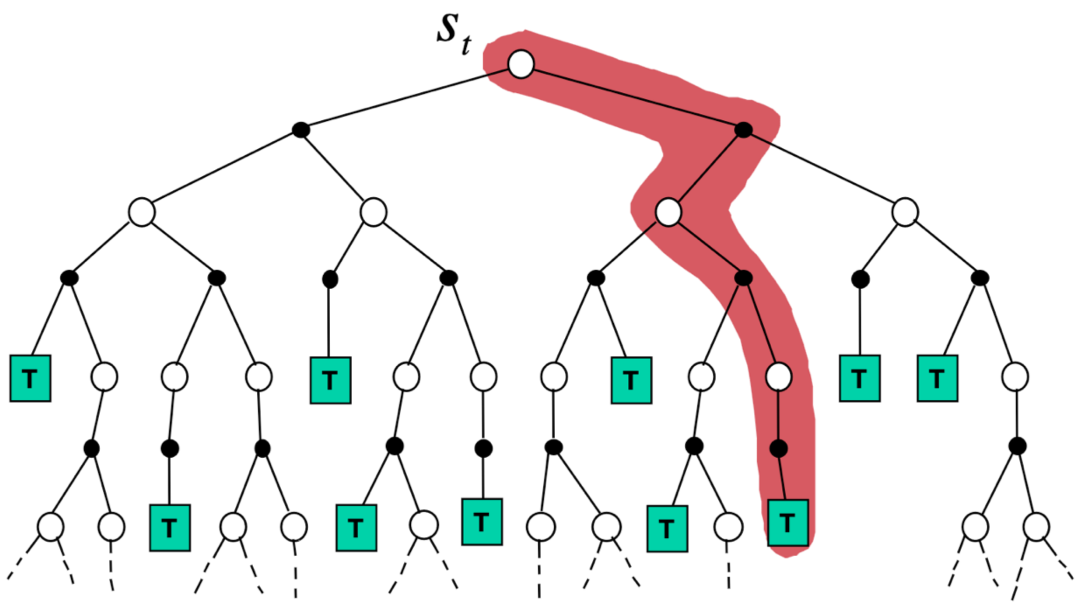
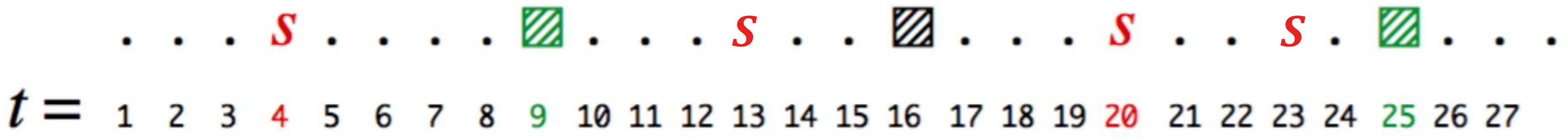


Image credit D. Silver

When to Update Value Estimation of a State

- Each green block denotes the terminal state in an episode
- State ***s*** might appear multiple times in an episode



First-visit MC prediction

Estimate $v_\pi(s)$ at the first visit to s in an episode

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:
- $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
- $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
- Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
    - Append G to $Returns(S_t)$
    - $V(S_t) \leftarrow \text{average}(Returns(S_t))$

Every-visit MC prediction

Estimate $v_\pi(s)$ at every visit to s in an episode

Every-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

~~Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:~~

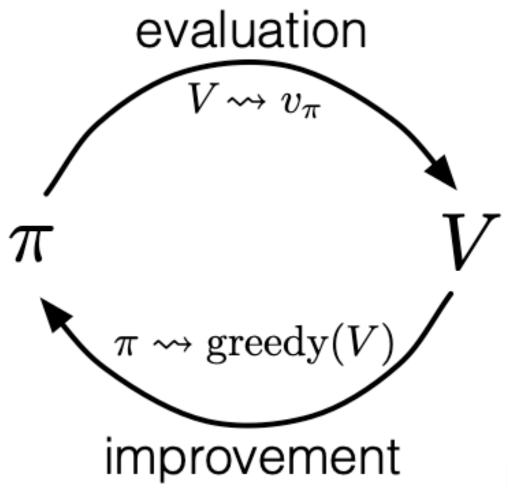
Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

How to Obtain Optimal Policies with Monte Carlo methods?

The same idea as generalized policy iteration: alternates optimization of policy evaluation and policy improvement

$$v(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v(s')]$$



But we don't know $p(s',r|s,a)$!

$$\pi(s) \leftarrow \underset{a}{argmax} \sum_{s',r} \sout{p(s',r\mid s,a)}[r + \gamma v_\pi(s')]$$

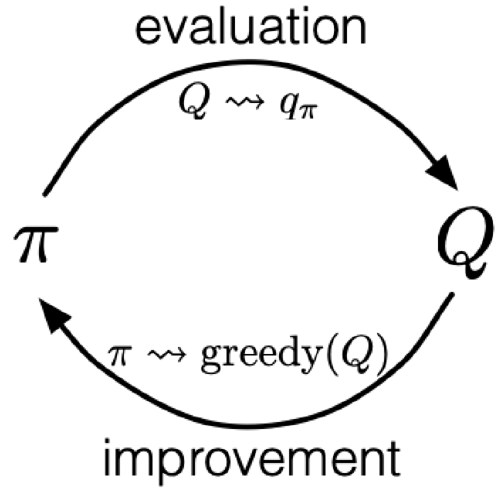
Slide adapted from K. Fragkiadaki

Convergence of Monte Carlo Control

Use state-action value $q_\pi(s,a)$, and we don't need to know $p(s',r|s,a)$!

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} q_*,$$

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$



$$\pi(s) \leftarrow \underset{a}{argmax}\, q_\pi(s,a)$$

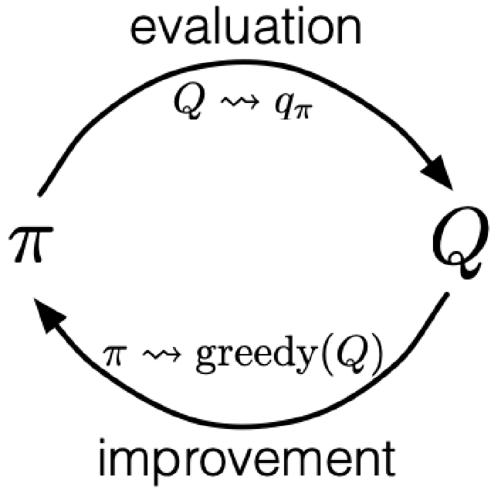
- MC methods converge, if:
  - We have infinite number of episodes (so the value estimate converges to the true value)
  - We visit every state-action pairs (so the value estimate will be the same as the true value)

Convergence of Monte Carlo Control

Use state-action value $q_\pi(s,a)$, and we don't need to know $p(s',r|s,a)$!

$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_*,$$

$$\begin{aligned} q_{\pi_k}(s,\pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s,\pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$



$$\pi(s) \leftarrow \underset{a}{argmax}\, q_\pi(s,a)$$

- MC methods converge, if:
  - We have infinite number of episodes (so the value estimate converges to the true value)
  - We visit every state-action pairs (so the value estimate will be the same as the true value)
- In other words, we need to explore!
  - If the policy always takes greedy action, we can never explore unseen state-action pairs

The Exploration-Exploitation Dilemma

- Exploitation: maximize the current highest reward: $\pi(s) = \underset{a}{argmax}\, q_\pi(s, a)$
- Exploration: maximize the information about the environment

The Exploration-Exploitation Dilemma

- **Exploitation**: maximize the current highest reward: $\pi(s) = \underset{a}{argmax}\, q_\pi(s, a)$
- **Exploration**: maximize the information about the environment

- Solutions:
  - <u>exploring starts</u>: Every state-action pair has a non-zero probability of being the starting pair
  - <u>ϵ-soft policies</u>: Most of the time choose the action with maximal estimated action values, but with probability ϵ select a random action
  - <u>off-policy</u>: use different policies for collecting experience and evaluating

Monte Carlo ES

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Monte Carlo ES

Converges to optimal policy, however, inefficient to start with every state-action pairs!

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

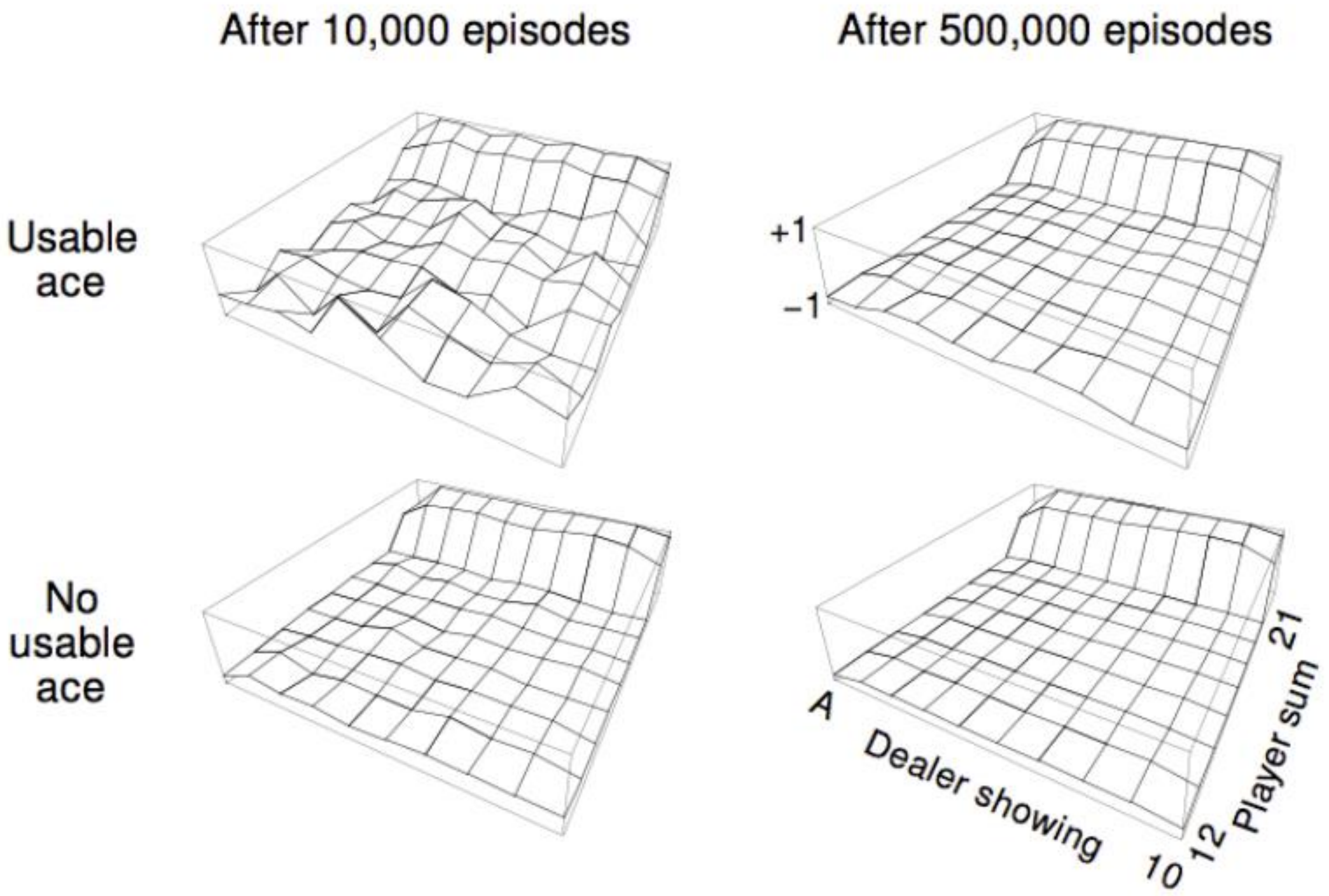
Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

The Blackjack Example

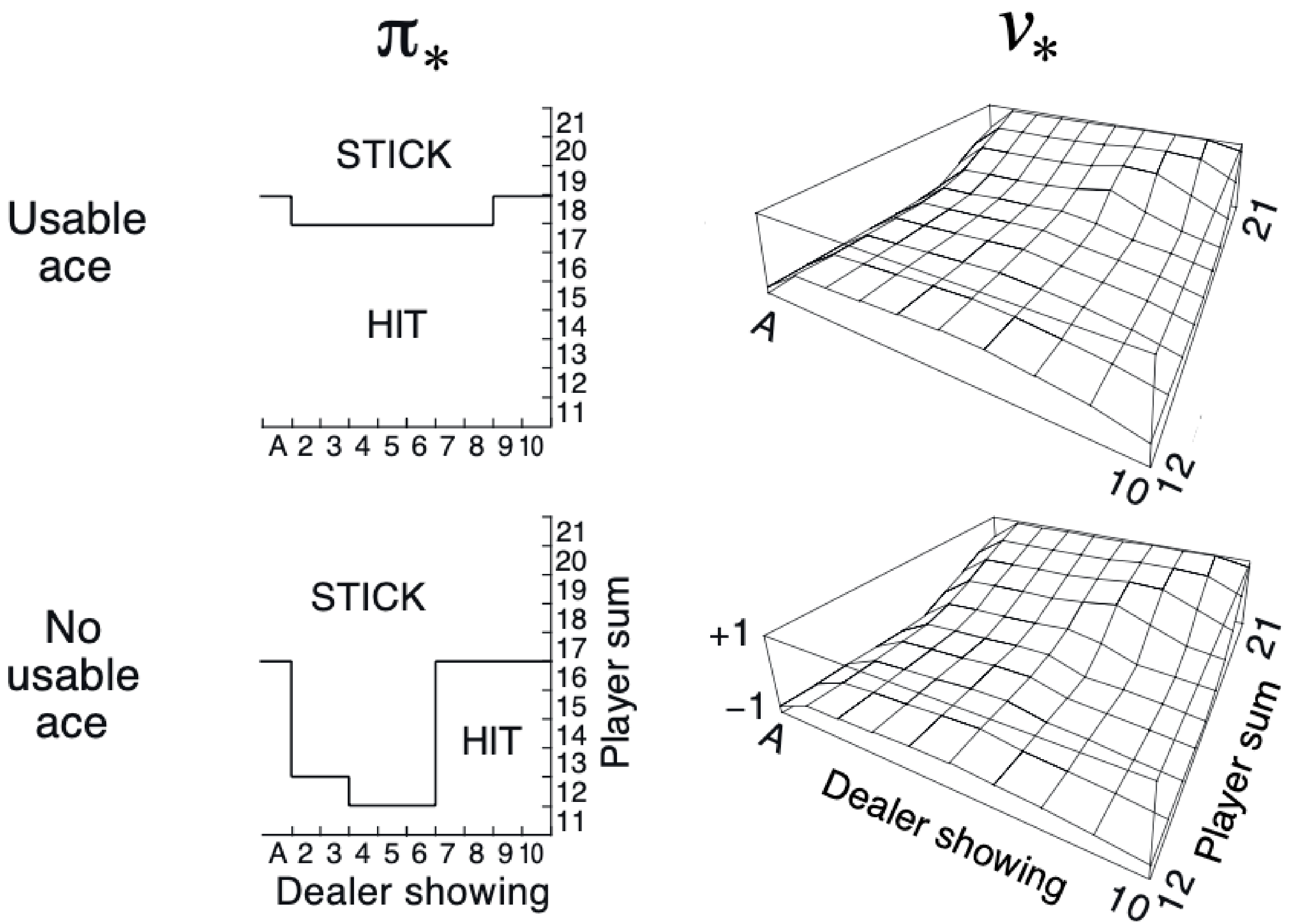
- States (200 of them):
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a "useable" ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - +1 if sum of cards > sum of dealer cards
  - 0 if sum of cards = sum of dealer cards
  - -1 if sum of cards < sum of dealer cards
- Reward for twist:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards < 12

Value Estimate by a Monte Carlo Methods



Policy: stick if sum of cards ≥ 20, otherwise twist

Optimal Policy found by Monte-Carlo ES



Slide credit D. Silver

Monte Carlo Control with ϵ-soft Policies

Explore with probability $\frac{\epsilon}{A(s)}$ and exploit with probability $1 - \epsilon + \frac{\epsilon}{A(s)}$

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

ϵ-soft Policies Improve the Original Policy

$$
\begin{aligned}
\mathbb{E}_{\pi'}\left[q_\pi\left(s,\pi'(s)\right)\right] &= \sum_a \pi'(a|s) q_\pi(s,a) \\
&= \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a) \;+\; (1-\varepsilon)\max_a q_\pi(s,a) \\
&\geq \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a) \;+\; (1-\varepsilon)\sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1-\varepsilon} q_\pi(s,a) \\
&= \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a) \;-\; \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a) \;+\; \sum_a \pi(a|s) q_\pi(s,a) \\
&= v_\pi(s).
\end{aligned}
$$

The Exploration-Exploitation Dilemma

- **Exploitation**: maximize the current highest reward: $\pi(s) = \underset{a}{argmax}\, q_{\pi}(s, a)$
- **Exploration**: maximize the information about the environment

- **ALL learning methods faces the dilemma**: learning state-action values conditions on subsequent optimal behaviors but they need to act sub-optimally to explore all state-action pairs

Both methods are compromises. They learn action values not for the optimal policy, but for a near-optimal policy that still explores.

- Solutions:
  - <u>exploring starts</u>: Every state-action pair has a non-zero probability of being the starting pair
  - <u>ϵ-soft policies</u>: Most of the time choose the action with maximal estimated action values, but with probability ϵ select a random action
  - <u>off-policy</u>: use different policies for collecting experience and evaluating

The Exploration-Exploitation Dilemma

- **Exploitation**: maximize the current highest reward: $\pi(s) = \underset{a}{argmax}\, q_\pi(s,a)$
- **Exploration**: maximize the information about the environment
- Solutions:

  Both methods are compromises. They learn action values not for the optimal policy, but for a near-optimal policy that still explores.

  - exploring starts: Every state-action pair has a non-zero probability of being the starting pair
  - ϵ-soft policies: Most of the time choose the action with maximal estimated action values, but with probability ϵ select a random action
  - off-policy: use different policies for collecting experience and evaluating
- **ALL learning methods faces the dilemma**: learning state-action values conditions on subsequent optimal behaviors but they need to act sub-optimally to explore all state-action pairs
  - Let's have two policies: policy b to explore, and policy π to behave optimally

On-Policy and Off-Policy Learning

- On-policy learning: learn v_π and q_π for policy π that executes and explores

- Off-policy learning: learn v_π and q_π for target policy π from experience collected by behavior policy b

- We only need coverage: every action taken under π is also taken, at least occasionally, under b

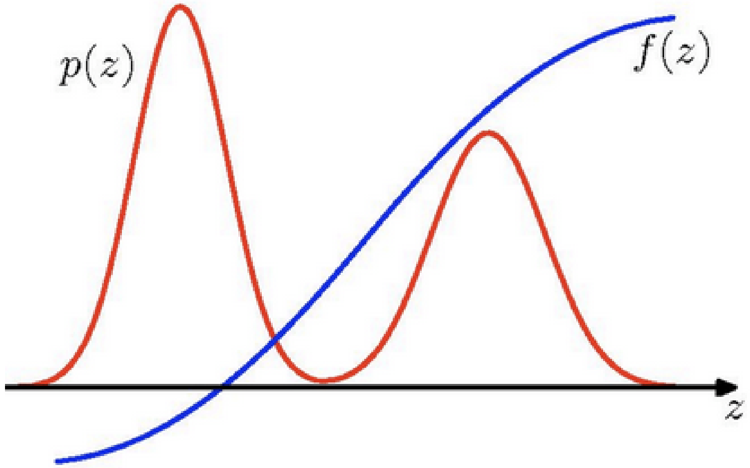
$$\pi(a|s) > 0 \;\; implies \;\; b(a|s) > 0$$

- The goodness of decoupling target and behavior policy:
  - ➢ Learn from observing humans or other agents
  - ➢ Re-use experience generated from old policies
  - ➢ Learn about optimal policy while following exploratory policy
  - ➢ Learn about multiple policies while following one policy

Slide adapted from D. Silver

Estimating Expectations

- General Idea: Draw independent samples $\{z^1, \ldots, z^n\}$ from distribution p(z) to approximate expectation:



$$\mathbb{E}[f] = \int f(z)p(z)dz \approx \frac{1}{N}\sum_{n=1}^{N} f(z^n) = \hat{f}.$$

Note that: $\mathbb{E}[f] = \mathbb{E}[\hat{f}].$

so the estimator has correct mean (unbiased).

- The variance: $$\text{var}[\hat{f}] = \frac{1}{N}\mathbb{E}\left[(f - \mathbb{E}[f])^2\right].$$

- Variance decreases as 1/N.

- **Remark**: The accuracy of the estimator does not depend on dimensionality of z.

Importance Sampling

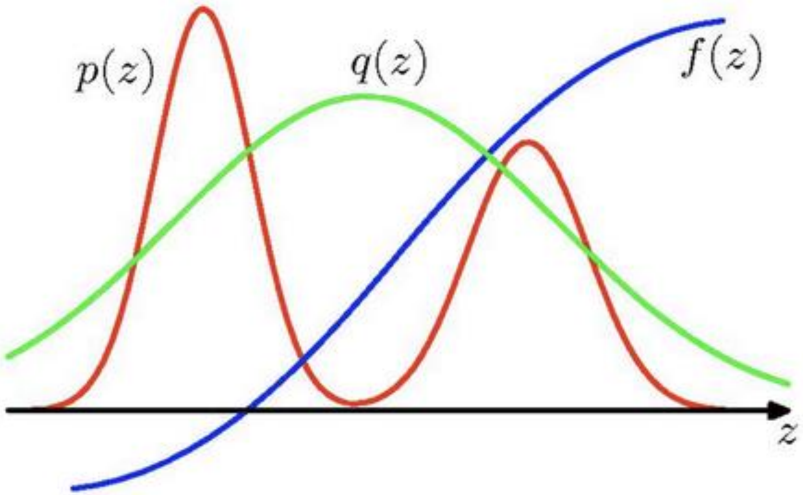
- Suppose we have an easy-to-sample proposal distribution q(z), such that

$$q(z) > 0 \text{ if } p(z) > 0.$$

$$\mathbb{E}[f] = \int f(z)p(z)dz$$

$$= \int f(z)\frac{p(z)}{q(z)}q(z)dz$$

$$\approx \frac{1}{N}\sum_n \frac{p(z^n)}{q(z^n)} f(z^n), \quad z^n \sim q(z).$$

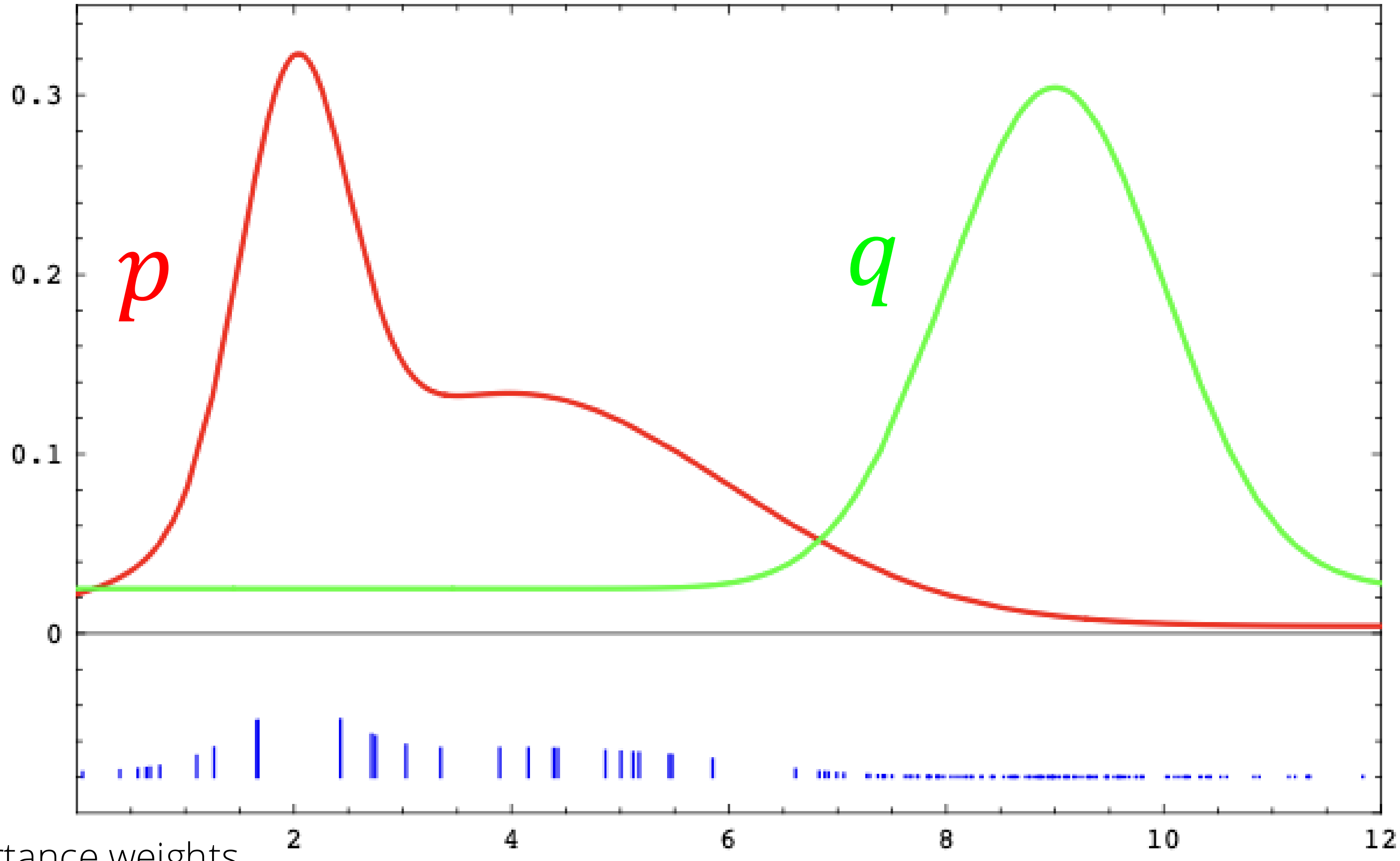


- The quantities

$$w^n = p(z^n)/q(z^n)$$

are known as importance weights.

- This is useful when we can evaluate the probability p but is hard to sample from it



importance weights

Image credit S. Thrun, W. Burgard and D. Fox

Importance Sampling Ratio

- The probability of state-action trajectory $A_t, S_t, \ldots, A_T, S_T$ under policy π:

$$
\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}
$$

- The importance sampling ratio between target policy π and behavior policy b:

$$
\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}
$$

- Estimate $v_\pi(s)$ from sampling with behavior policy b

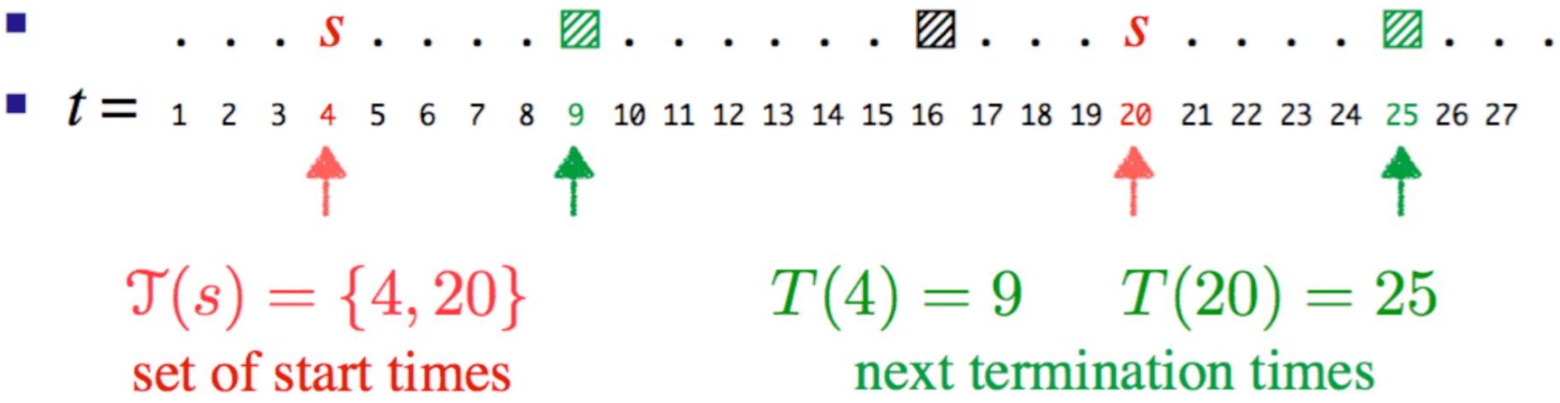
$$
\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)
$$

Importance Sampling

- Ordinary importance sampling forms estimate

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}.$$

- New notation: time steps increase across episode boundaries:



Importance Sampling

- Two ways of averaging weighted returns:
  - Ordinary importance sampling forms estimate:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}.$$

  - Weighted importance sampling forms estimate:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Ordinary vs. Weighted Importance Sampling

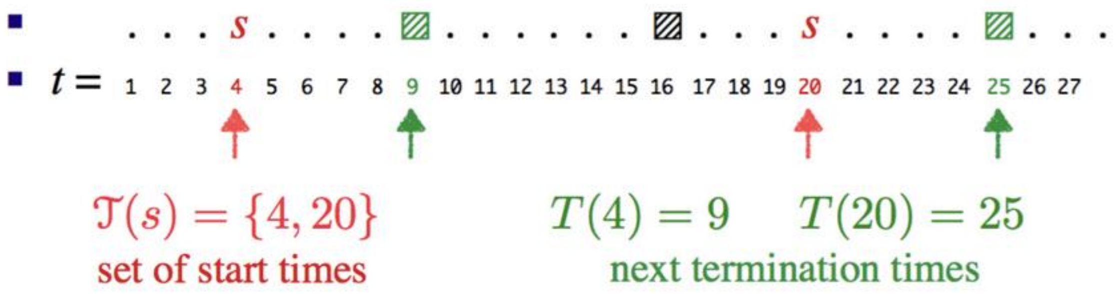
Ordinary Sampling:

First time of termination following time t

return after t up through T(t)

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}.$$

Every time: the set of all time steps in which state s is visited



Weighted Sampling:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

First-visit MC:

- Ordinary Sampling is unbiased, but the variance is unbounded
- Weighted Sampling is biased, but with much lower variance

Every-visit MC:

- Ordinary Sampling is biased
- Weighted Sampling is biased

Proof: https://link.springer.com/article/10.1007/BF00114726

Image credit K. Fragkiadaki

Importance Sampling

- Two ways of averaging weighted returns:
  - Ordinary importance sampling forms estimate:

Correct mean of value following policy π

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}.$$

  - Weighted importance sampling forms estimate:

These ratio in the numerator is cancelled by the denominator. The estimation is not correct anymore....

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Remember the Incremental Implementation in the Bandit Problem

- Let Q_n denote the estimate of its action value after being selected $n-1$ times

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

- Let's start rewriting Q_n:

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big],
\end{aligned}$$

Incremental Implementation of Ordinary Importance Sampling

- We have:

$$V_n(s) = \frac{\rho_1 G_1 + \cdots \rho_{n-1} G_{n-1}}{n-1}$$

- We can re-write $V(s)$ as:

$$V_{n+1}(s) = V_n(s) + \frac{1}{n}[\rho_n G_n - V_n(s)]$$

Incremental Implementation of Weighted Importance Sampling

- We can re-write weighted importance sampling as:

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \qquad n \geq 2, \qquad W_i = \rho_{t_i:T(t_i)-1}$$

- The update rule is then:

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}\Big[G_n - V_n\Big], \qquad n \geq 1,$$

$$C_{n+1} \doteq C_n + W_{n+1},$$

Off-Policy Monte Carlo Prediction

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

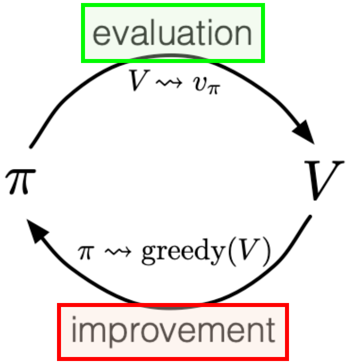
Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Off-policy Monte Carlo Control

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \in \mathbb{R}$ (arbitrarily)
  $C(s,a) \leftarrow 0$
  $\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
  $b \leftarrow$ any soft policy
  Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
    $\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (with ties broken consistently)
    If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
    $W \leftarrow W \frac{1}{b(A_t|S_t)}$



Only learns from the tails of episodes (when all actions are greedy). Learning is very slow!

Check Section 5.8 and 5.9 of “Reinforcement Learning: An Introduction” for Discounting-aware Importance Sampling

Quick Summary: DP vs. MC

- Dynamic Programming (DP) methods are efficient, which bootstrap value functions from existing estimates.

$$V(S_t) \leftarrow \sum_{A_t} \pi(A_t|S_t) \sum_{S_{t+1},R_{t+1}} p(S_{t+1}, R_{t+1}|S_t, A_t)[R_{t+1} + \gamma V(S_{t+1})]$$

- Monte Carlo (MC) methods: must wait until the end of the episode to learn value functions (only when the return is known)

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

Temporal-Difference Learning

- Temporal-Difference (TD) methods: combine Monte Carlo methods with Dynamic Programming methods that wait only until the next time step and bootstrap value functions from existing estimates

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

- ➢ TD target: $R_{t+1} + \gamma V(S_{t+1})$
- ➢ TD error: $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

Backup Diagram for Temporal-Difference Methods

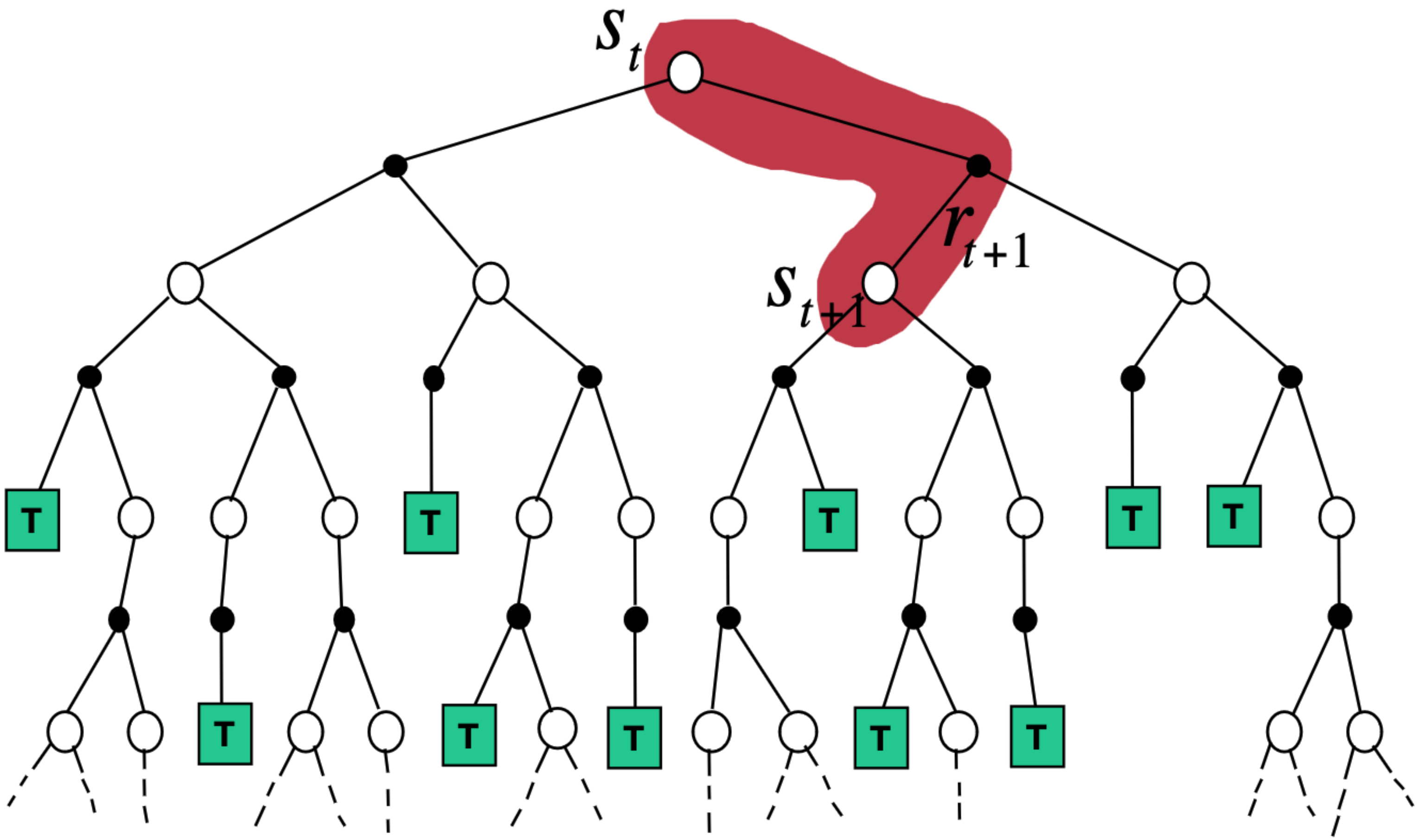
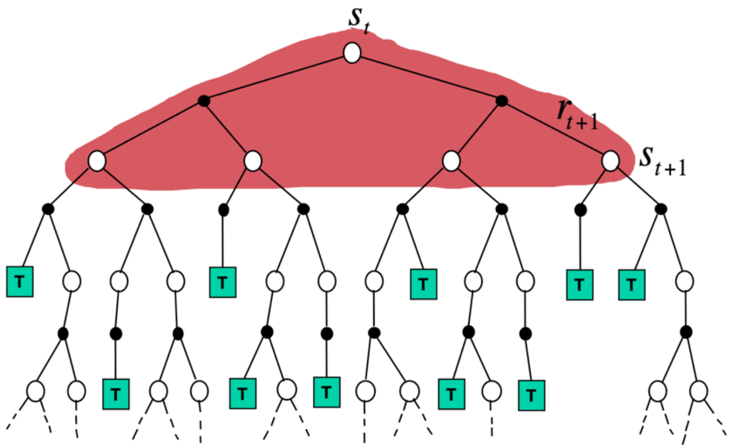


Image credit D. Silver

Backup diagram: DP vs. MC vs. TD

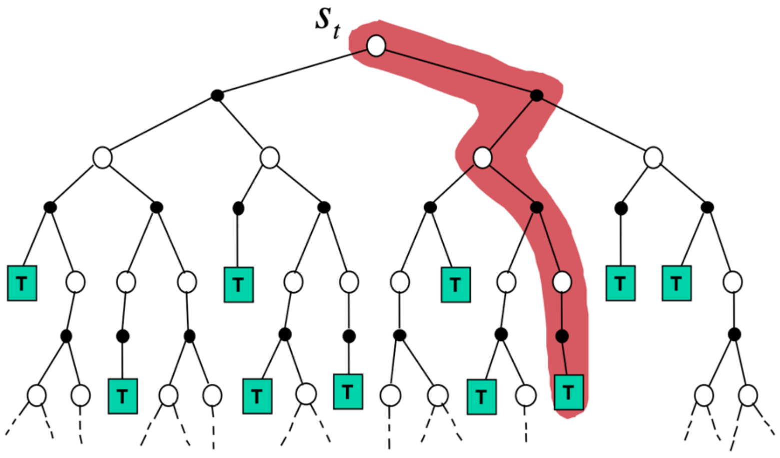
Dynamic Programming Backup



$V(S_t) \leftarrow \sum_{A_t} \pi(A_t|S_t) \sum_{S_{t+1},R_{t+1}} p(S_{t+1},R_{t+1}|S_t,A_t)[R_{t+1} + \gamma V(S_{t+1})]$

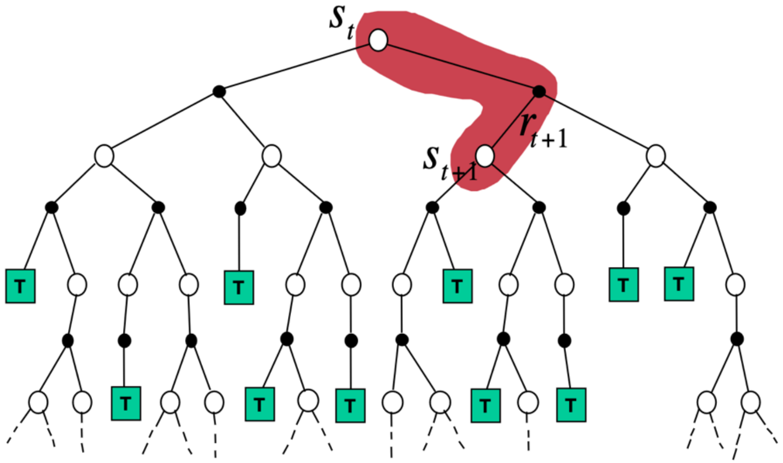
Expected updates: based on the summation of all successors

MC Backup



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

TD Backup



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

Sample updates: based on a single sample successor

Image credit D. Silver

DP vs. MC vs. TD

	Bootstrap	Sample
Dynamic Programming	✓	×
Monte Carlo	×	✓
Temporal Difference	✓	✓

DP vs. MC vs. TD

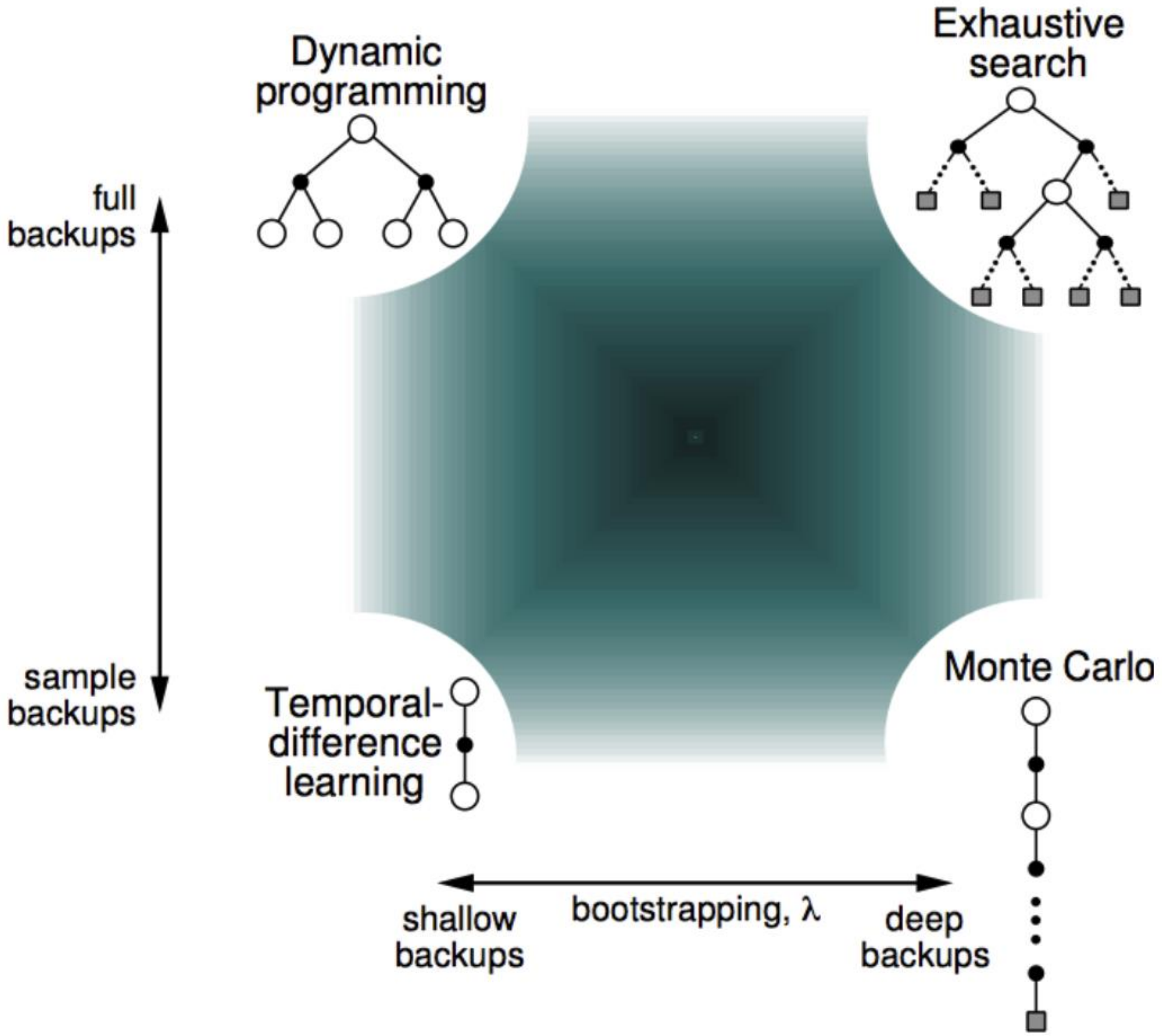


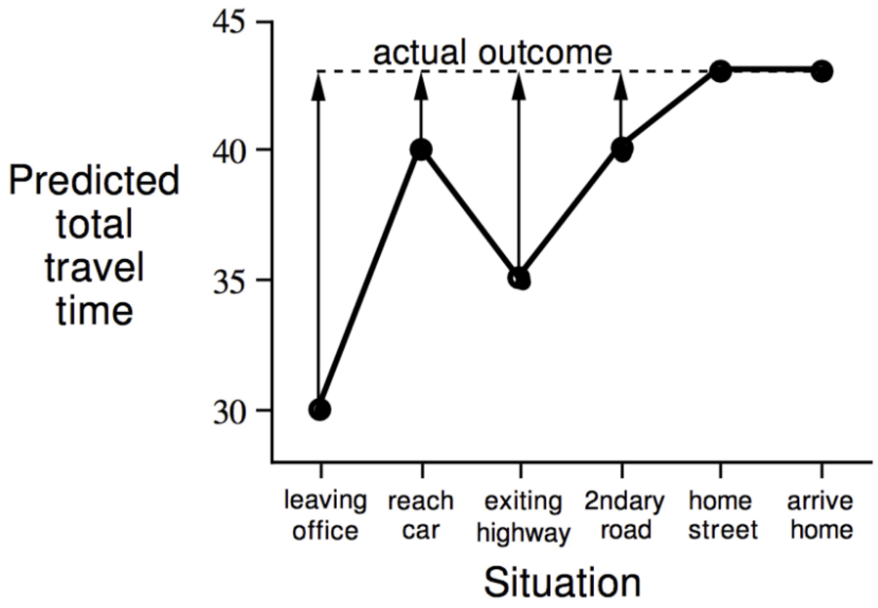
Image credit D. Silver

The Driving Home Example

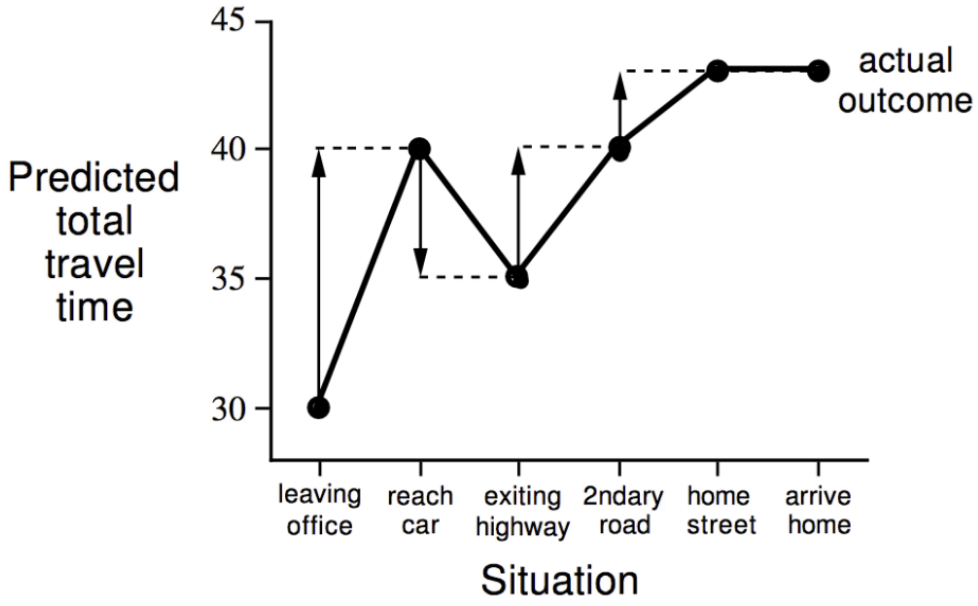
State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

The Driving Home Example

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



Nice Properties of TD

- TD can learn before knowing the final outcome
  - ➢ TD can learn online after every step
  - ➢ MC must wait until end of episode before return is known

- TD can learn without the final outcome
  - ➢ TD can learn from incomplete sequences
  - ➢ MC can only learn from complete sequences
  - ➢ TD works in continuing (non-terminating) environments
  - ➢ MC only works for episodic (terminating) environments

- Both TD and MC converge (under certain conditions). Which converge better / faster?

Slide credit K. Fragkiadaki and D. Silver

Tabular TD(0) Prediction

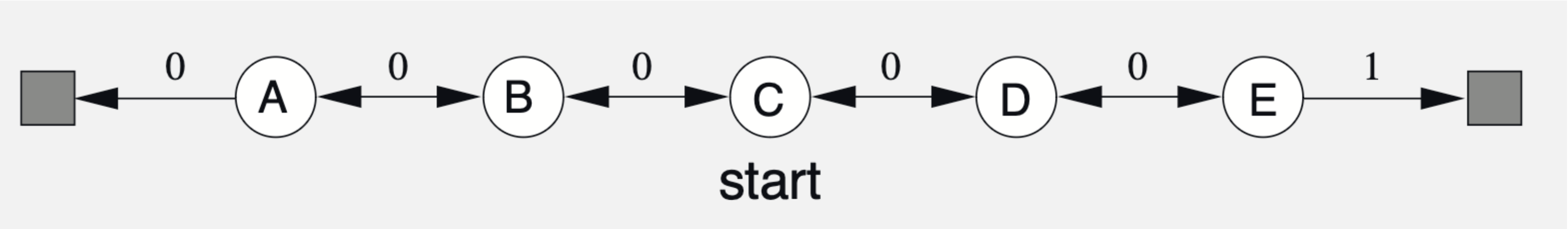
Tabular TD(0) for estimating v_π

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

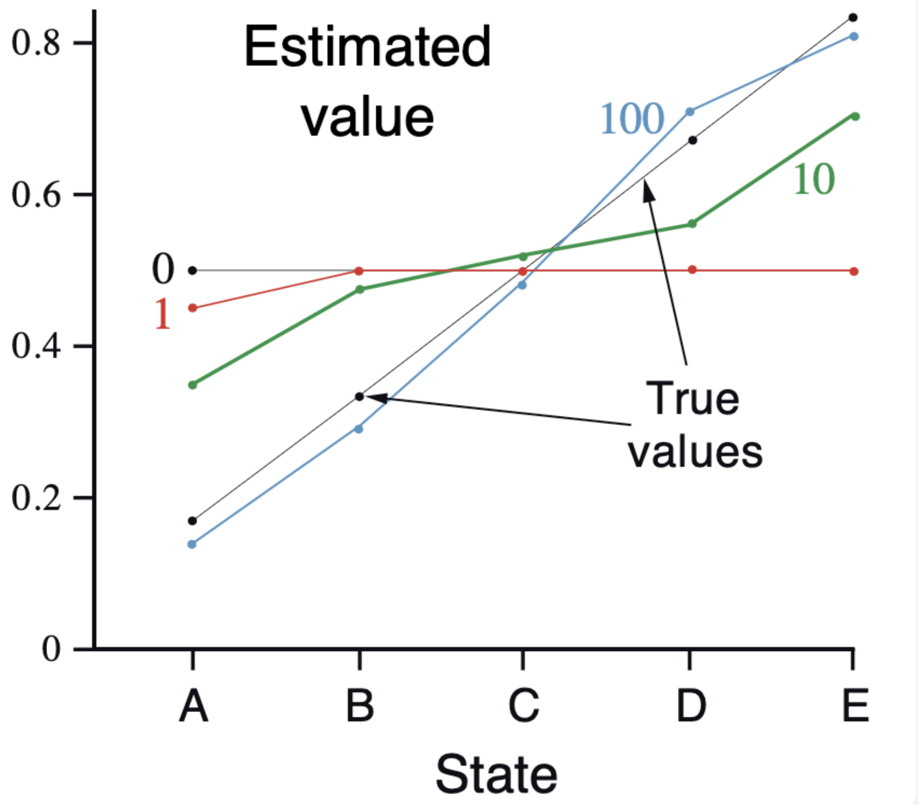
Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

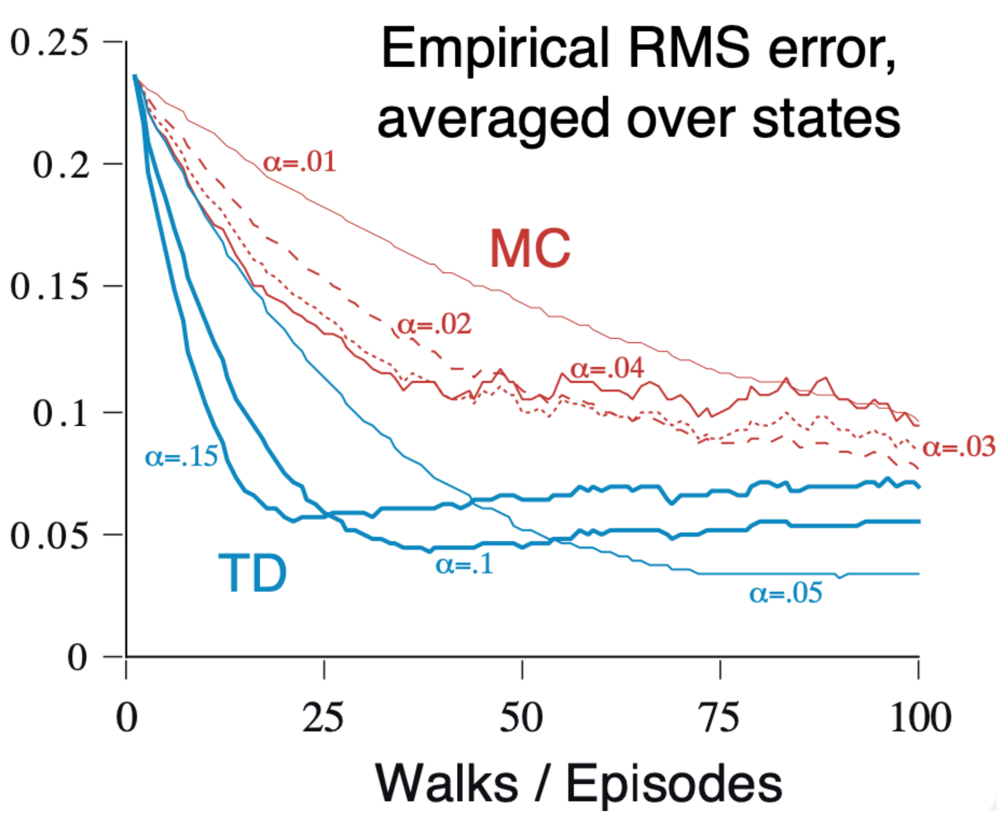
Random Walk Example

- Assume we have a Markov Reward Process—a MDP without actions.
- Always start from the center, moving to the left / right with equal probability
- The ground-truth values of state A to E are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$
- Does TD(0) converge faster and better than MC?



Random Walk Example



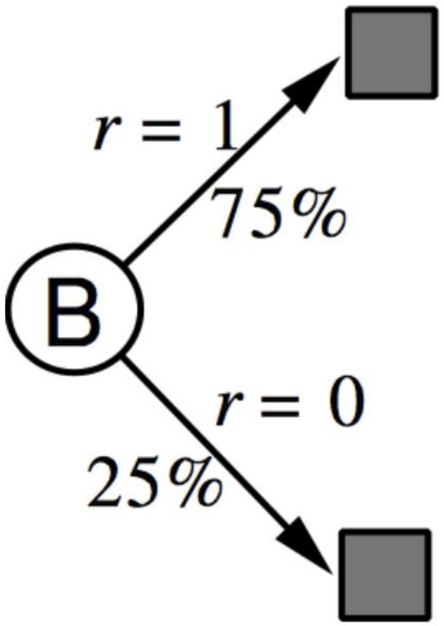


A-B Example

Two states A, B; no discounting; 8 episodes of experience

$\mathrm{A}, 0, \mathrm{B}, 0$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 0$

What is $V(A), V(B)$?



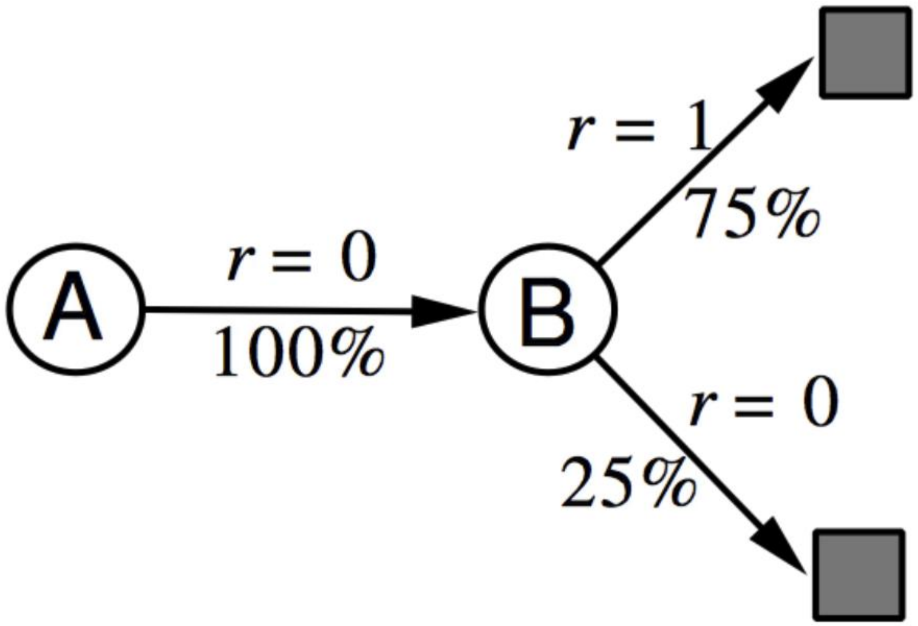
Batch updating: Repeatedly train on episodes until convergence.

$$V(B) = \frac{1}{8}(0 + 1 + 1 + 1 + 1 + 1 + 1 + 0) = 0.75$$

A-B Example

Two states A, B; no discounting; 8 episodes of experience

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



What is $V(A), V(B)$?

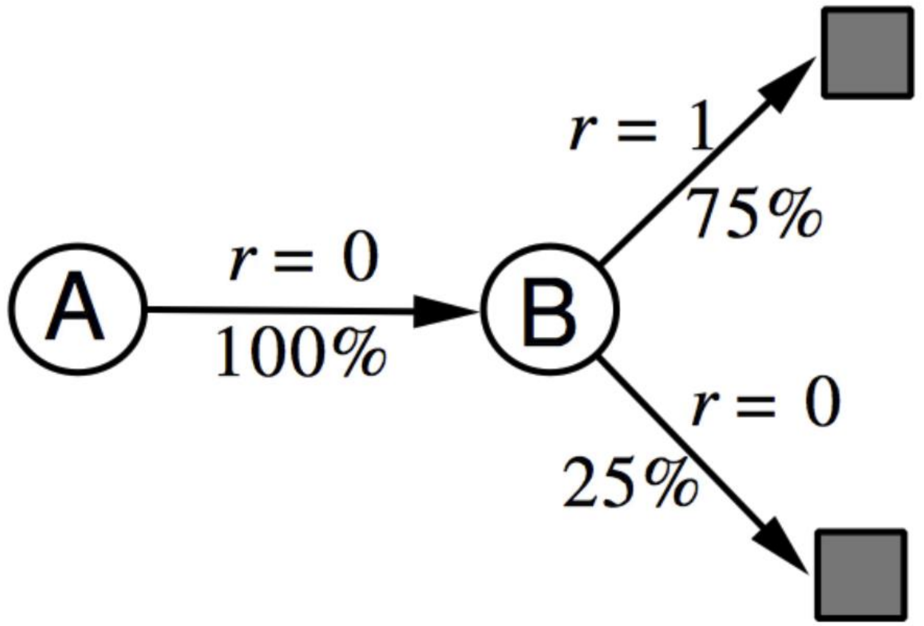
For MC methods:.

$$V(A) = R_t + \cdots + R_T = 0 + 0 = 0$$

A-B Example

Two states A, B; no discounting; 8 episodes of experience

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



What is $V(A), V(B)$?

For MC methods:.

$$V(A) = R_t + \cdots + R_T = 0 + 0 = 0$$

For TD methods:

$$V(A) = R_t + V(B) = 0 + 0.75 = 0.75$$

Why Does TD Converge Better than MC?

- Monte Carlo in batch setting converges to min MSE (mean squared error)
  - Minimize loss with respect to observed returns
  - In AB example, $V(A) = 0$
- TD(0) converges to DP policy V^π for the MDP with the maximum likelihood model estimates
- **Aka same as dynamic programming with certainty equivalence!**
  - Maximum likelihood Markov decision process model

$$\hat{P}(s'|s,a) = \frac{1}{N(s,a)} \sum_{k=1}^{i} \mathbb{1}(s_k = s, a_k = a, s_{k+1} = s')$$

$$\hat{r}(s,a) = \frac{1}{N(s,a)} \sum_{k=1}^{i} \mathbb{1}(s_k = s, a_k = a) r_k$$

  - Compute V^π using this model
  - In AB example, $V(A) = 0.75$

Slide credit E. Brunskill

Bias and Variance Analysis

- Return $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:
    - Return depends on *many* random actions, transitions, rewards
    - TD target depends on *one* random action, transition, reward

Biased Estimation of TD methods

$$
\begin{aligned}
V_{n+1}(s) &= V_n(s) + \alpha[R + V(s') - V_n(s)] \\
&= (1-\alpha)V_n(s) + \alpha(R + V(s')) \\
&= (1-\alpha)[V_{n-1}(s) + \alpha[R + V(s') - V_{n-1}(s)]] + \alpha(R + V(s')) \\
&= \dots \\
&= (1-\alpha)^n V_1(s) + \sum_{i=1}^{n} \alpha(1-\alpha)^{i-1}(R + V(s'))
\end{aligned}
$$

Biased by $V_1(s)$

https://stats.stackexchange.com/questions/454856/why-is-temporal-difference-learning-biased-in-reinforcement-learning

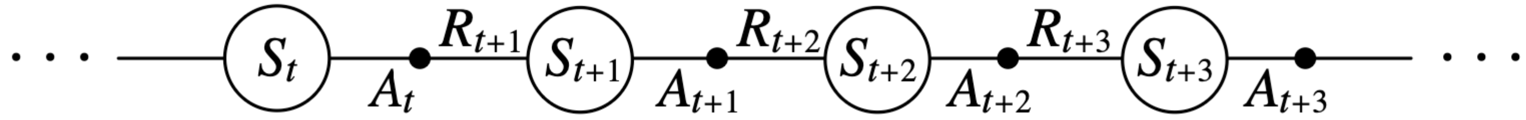
Bias and Variance Analysis

- MC has high variance, zero bias
    - Good convergence properties
    - (even with function approximation)
    - Not very sensitive to initial value
    - Very simple to understand and use
- TD has low variance, some bias
    - Usually more efficient than MC
    - TD(0) converges to $v_\pi(s)$
    - (but not always with function approximation)
    - More sensitive to initial value

Sarsa: On-policy TD Control

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

- We can learn an action-value function in a similar manner as a state-value function. Instead of considering transitions from state to state, we now consider transitions from state-action pair to state-action pair



Sarsa: On-policy TD Control

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

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 $Q(terminal, \cdot) = 0$

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

Expected Sarsa

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

Eliminate variance due to the random selection of A_t

Expected Sarsa:

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

3. Loop for each episode:
4.   Initialize S
5.   Loop for each step of episode:
6.   Choose A from S using policy derived from Q (e.g., ε-greedy)
7.   Take action A, observe R, S'
8. $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \sum_a \pi\left(a \mid S_{t+1}\right) Q\left(S_{t+1}, a\right) - Q(S, A)\right]$
9. $S \leftarrow S'$
10.   until S is terminal

Q-learning: Off-policy TD Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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

Q-learning: Off-policy TD Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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 $Q(terminal, \cdot) = 0$

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

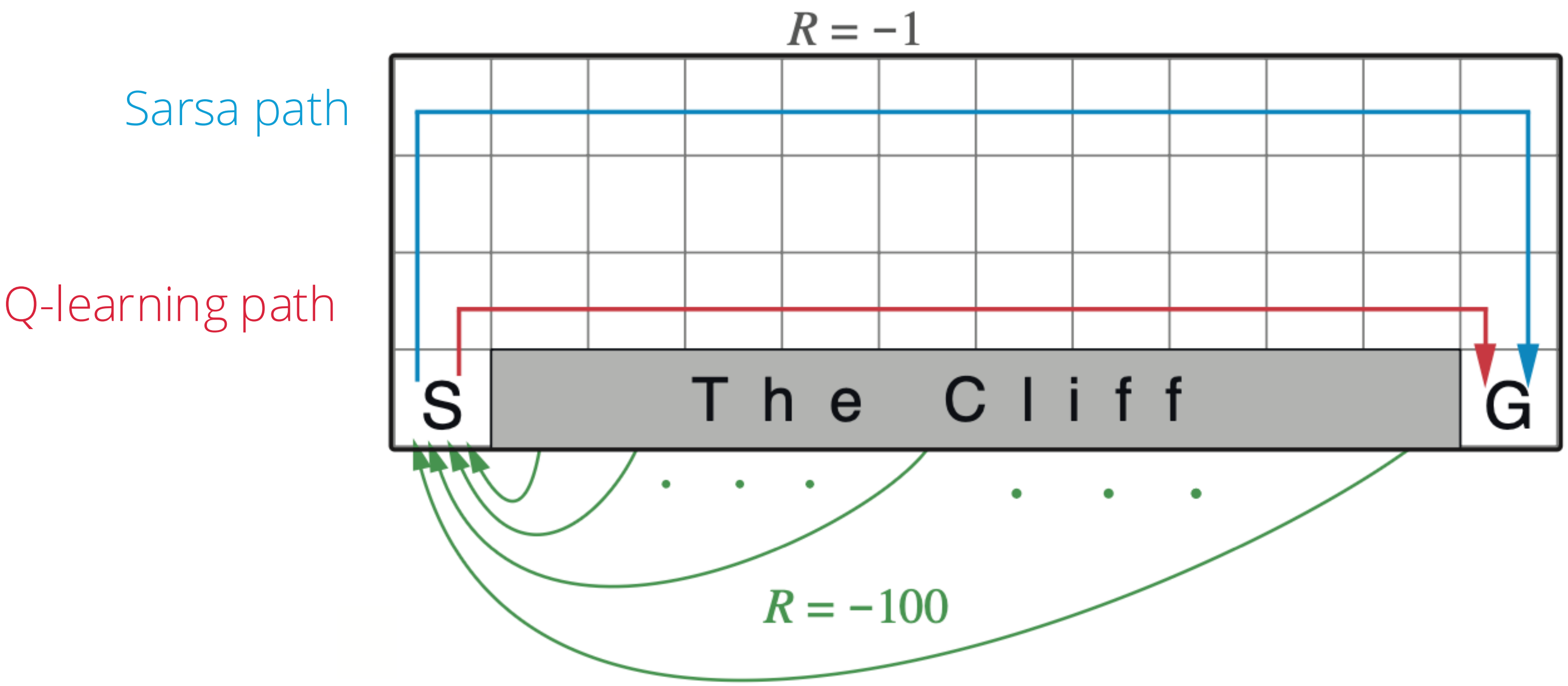
behavior policy

target policy

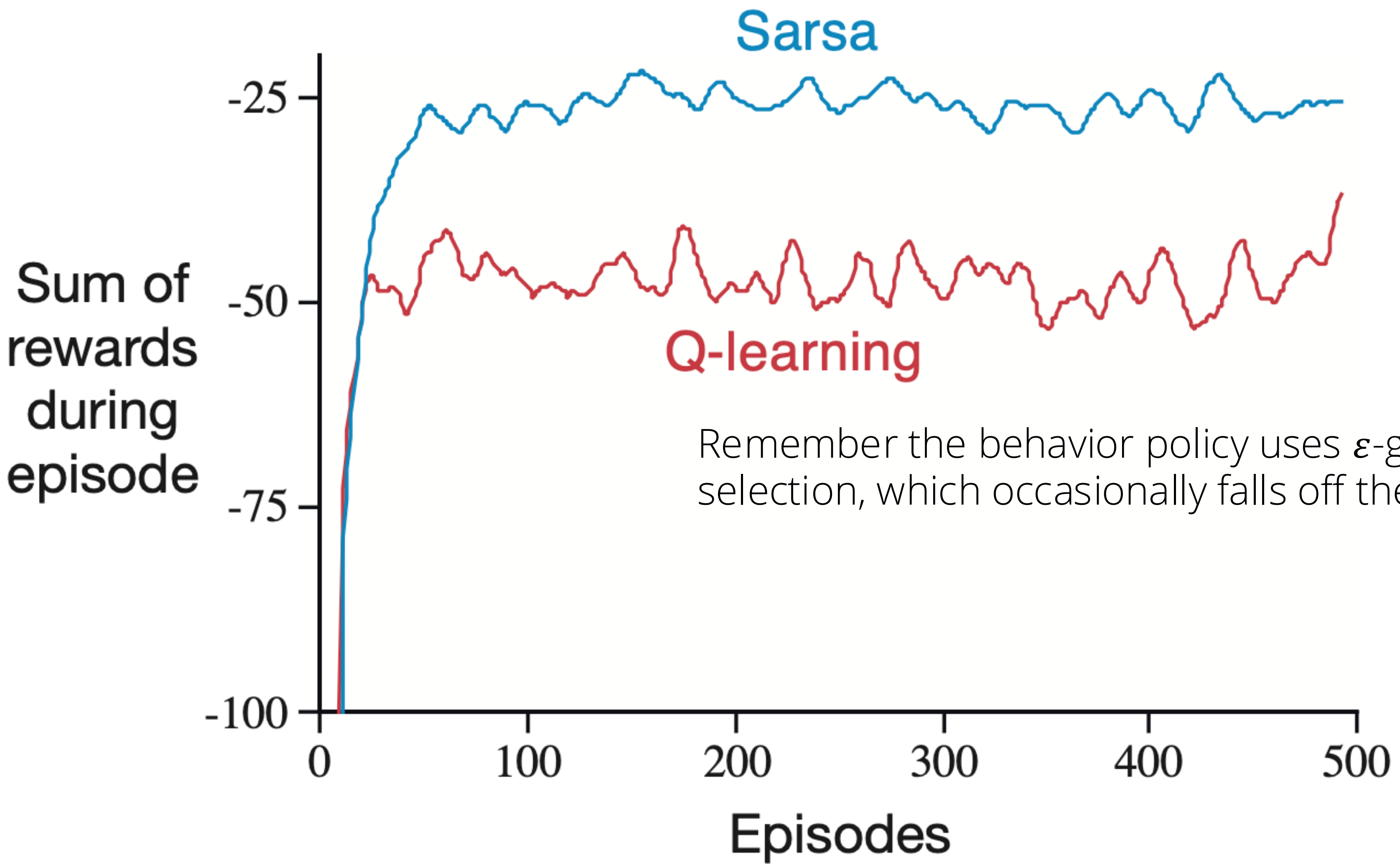
- The learned action-value function approximates q^*
- If all state-action pairs continue to be updated, Q has been shown to converge with probability 1 to q^*

Cliff Walking Example

- The behavior policy uses ε-greedy action selection, with $\varepsilon = 0.1$
- Action: up, down, left and right
- Reward is -100 at the Cliff region, otherwise, reward is -1



Cliff Walking Example



Remember the behavior policy uses ε-greedy action selection, which occasionally falls off the cliff!

Maximization Bias and Double Q-Learning

- The estimated values $Q(s, a)$ are often uncertain and distributed some above and some below zero. The maximum of estimated values induces a positive bias.

- Let say the true values of state s and many actions a are all zero, but estimated values $Q(s, a)$ has positive bias

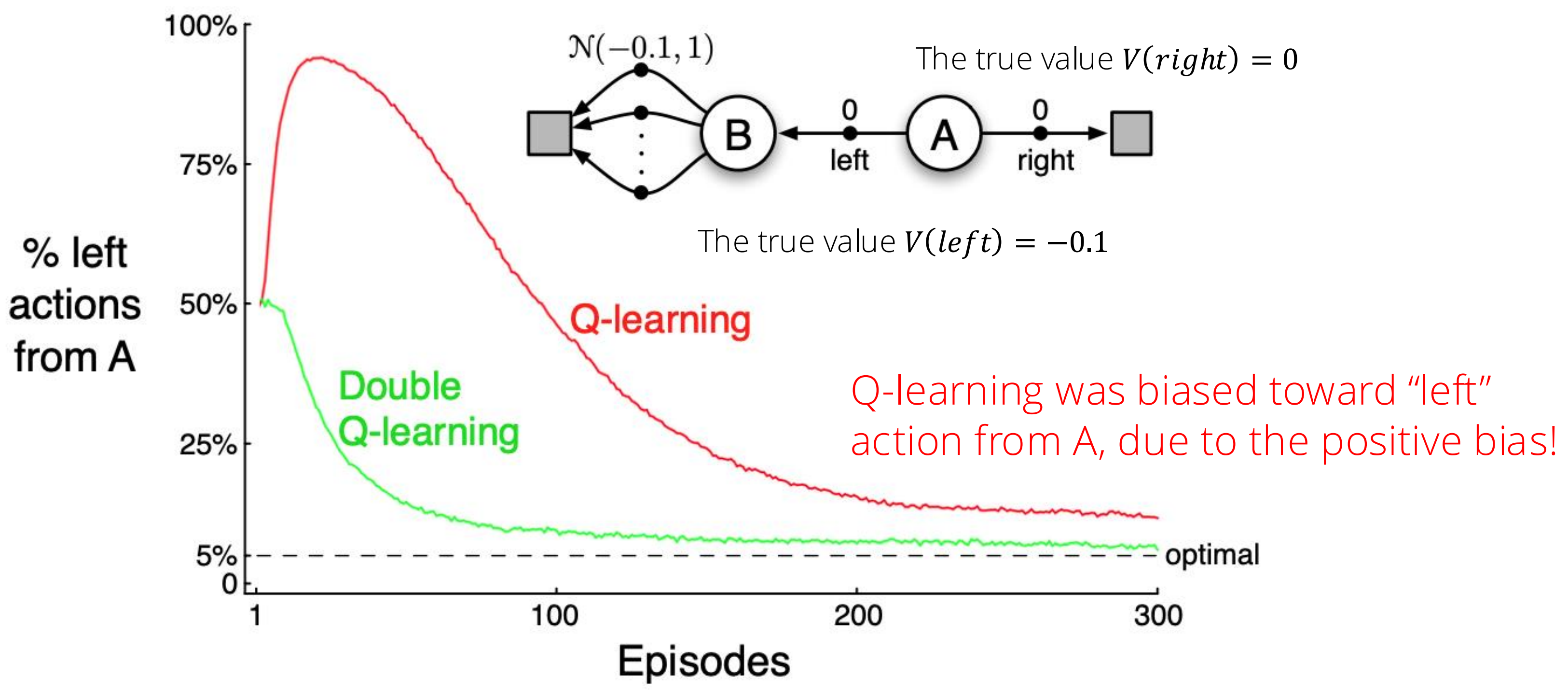
positive bias is introduced by the "maximum" operator

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

- This is because we use the same samples to determine the maximizing action and to estimate is values!

Maximization Bias Example

- Action: left and right
- Reward is 0 when transitioning from A to B; reward is drawn from $\mathcal{N}(-0.1,1)$ when transitioning from B to left.
- Taking "left" action from A should always be worse than "right" action



Double Q-Learning

- The estimated values $Q(s, a)$ are often uncertain and distributed some above and some below zero. The maximum of estimated values induces a positive bias.

- This is because we use the same samples to determine the maximizing action and to estimate is values!

- Solution: use two sets of samples to learn two independent estimates Q_1 and Q_2
  - Q_1 determines the maximizing action:

$$A^* = \underset{a}{argmax}\, Q_1(s, a)$$

  - Q_2 provides the estimate of its value:

$$Q_2(s, A^*) = Q_2(s, \underset{a}{argmax}\, Q_1(s, a))$$

Double Q-Learning

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q_2\big(S_{t+1}, \operatorname*{arg\,max}_a Q_1(S_{t+1}, a)\big) - Q_1(S_t, A_t)\Big].$$

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
        Take action A, observe R, S'
        With 0.5 probabilility:
            $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \operatorname{arg\,max}_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
        else:
            $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \operatorname{arg\,max}_a Q_2(S', a)\big) - Q_2(S, A)\Big)$
        $S \leftarrow S'$
    until S is terminal

Quick Recap: Temporal-Difference Learning

- Temporal-Difference (TD) methods: combine Monte Carlo methods with Dynamic Programming methods that wait only until the next time step and bootstrap value functions from existing estimates

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

We call this formulation 1-step TD
We can also have n-step TD

N-step TD Prediction

- n-step TD:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}) - V(S_t)]$$

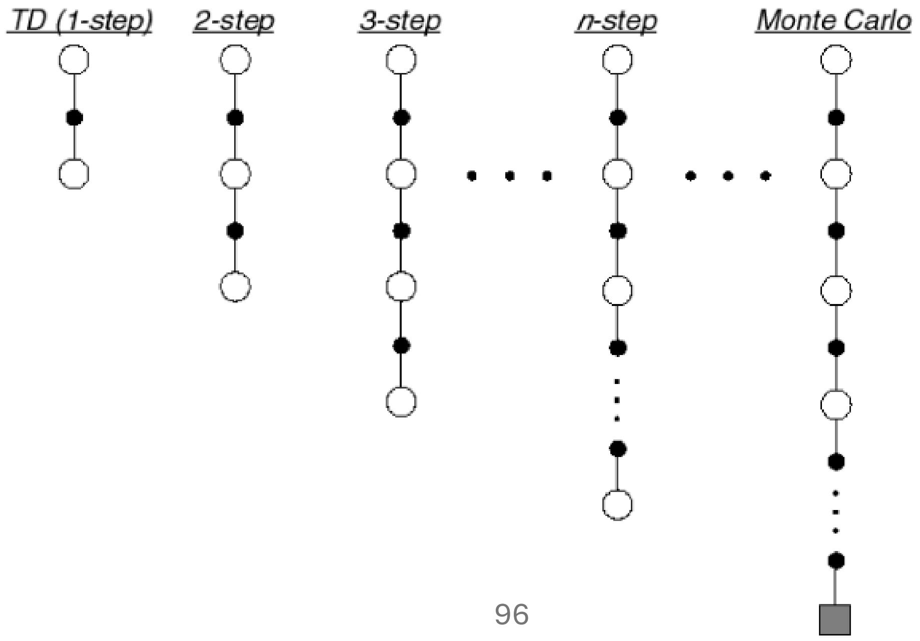
N-step TD Prediction

- n-step TD:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}) - V(S_t)]$$

- When $n \to \infty$, n-step TD becomes an MC method:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T - V(S_t)]$$



N-step TD Prediction

n-step TD for estimating $V \approx v_\pi$

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n+1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$:
    | If $t < T$, then:
    | Take an action according to $\pi(\cdot|S_t)$
    | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    | If S_{t+1} is terminal, then $T \leftarrow t+1$
    | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
    | If $\tau \geq 0$:
    | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
    | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
    | $V(S_\tau) \leftarrow V(S_\tau) + \alpha \left[G - V(S_\tau)\right]$
    Until $\tau = T - 1$

No bootstrapping until time step $t + n$

N-step TD Prediction

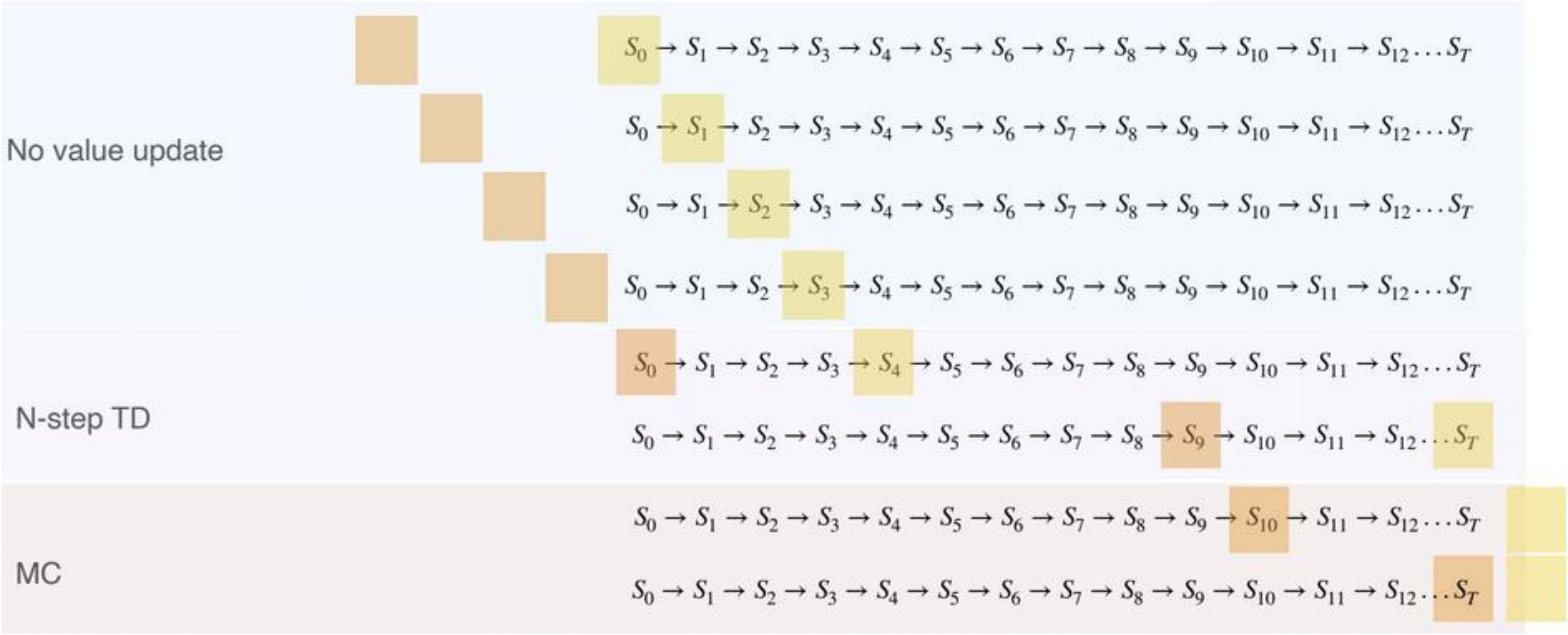


Image credit K. Fragkiadaki

On-policy n-step Action-Value Methods

- Action-value form of n-step return

$$G_{t:t+n} \doteq R_{t+1}+\gamma R_{t+2}+\cdots+\gamma^{n-1}R_{t+n}+\gamma^n Q_{t+n-1}(S_{t+n},A_{t+n}),\ \ n\geq 1, 0\leq t< T-n,$$

- n-step Sarsa

$$Q_{t+n}(S_t,A_t) \doteq Q_{t+n-1}(S_t,A_t)+\alpha\left[G_{t:t+n}-Q_{t+n-1}(S_t,A_t)\right]$$

- n-step expected Sarsa

$$G_t^{(n)} \doteq R_{t+1}+\cdots+\gamma^{n-1}R_{t+n}+\gamma^n\sum_a \pi(a|S_{t+n})Q_{t+n-1}(S_{t+n},a)$$

Off-policy n-step Action-Value Methods

- Importance-sampling ratio

$$\rho_{t:h} \doteq \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

- Weighted estimated value functions with importance-sampling ratio
- Off-policy n-step TD

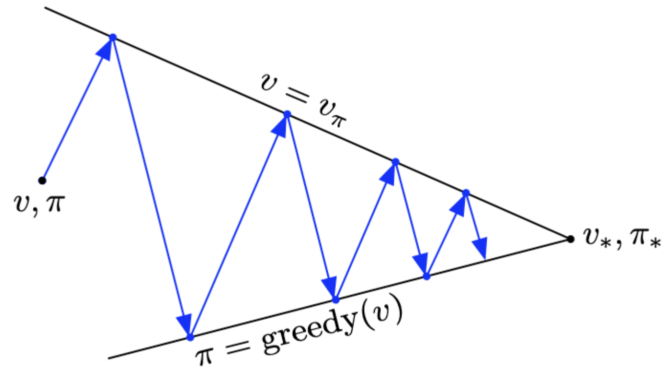
$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \rho_{t:t+n-1} \left[G_{t:t+n} - V_{t+n-1}(S_t) \right], \qquad 0 \leq t < T,$$

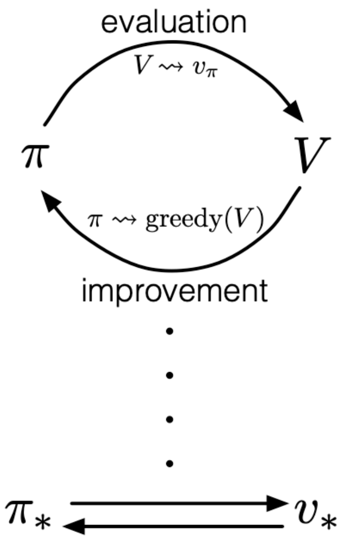
- Off-policy n-step Sarsa

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n} \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t) \right]$$

Summary

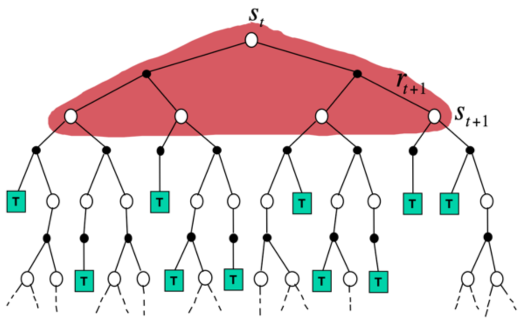
Generalized Policy Iteration



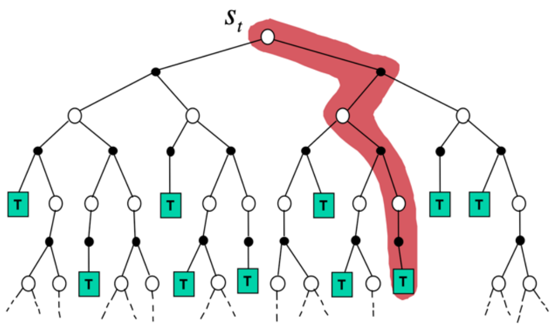


DP vs. MC vs. TD

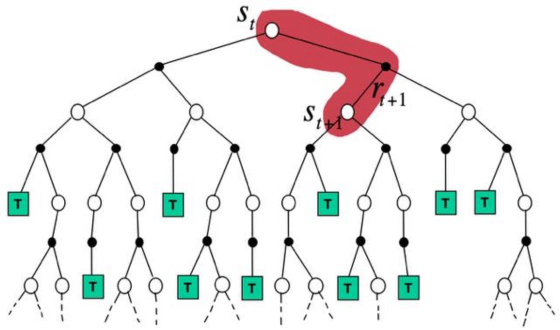
Dynamic Programming Backup



MC Backup



TD Backup



	Bootstrap	Sample
DP	✓	×
MC	×	✓
TD	✓	✓

- DP: $V(S_t) \leftarrow \sum_{A_t} \pi(A_t|S_t) \sum_{S_{t+1},R_{t+1}} p(S_{t+1}, R_{t+1}|S_t, A_t)[R_{t+1} + \gamma V(S_{t+1})]$
- MC: $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$
- TD: $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

- **On-policy learning**: learn value and execute with the same policy
- **Off-policy learning**: learn and execute with different policies

Importance Sampling

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k) p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$