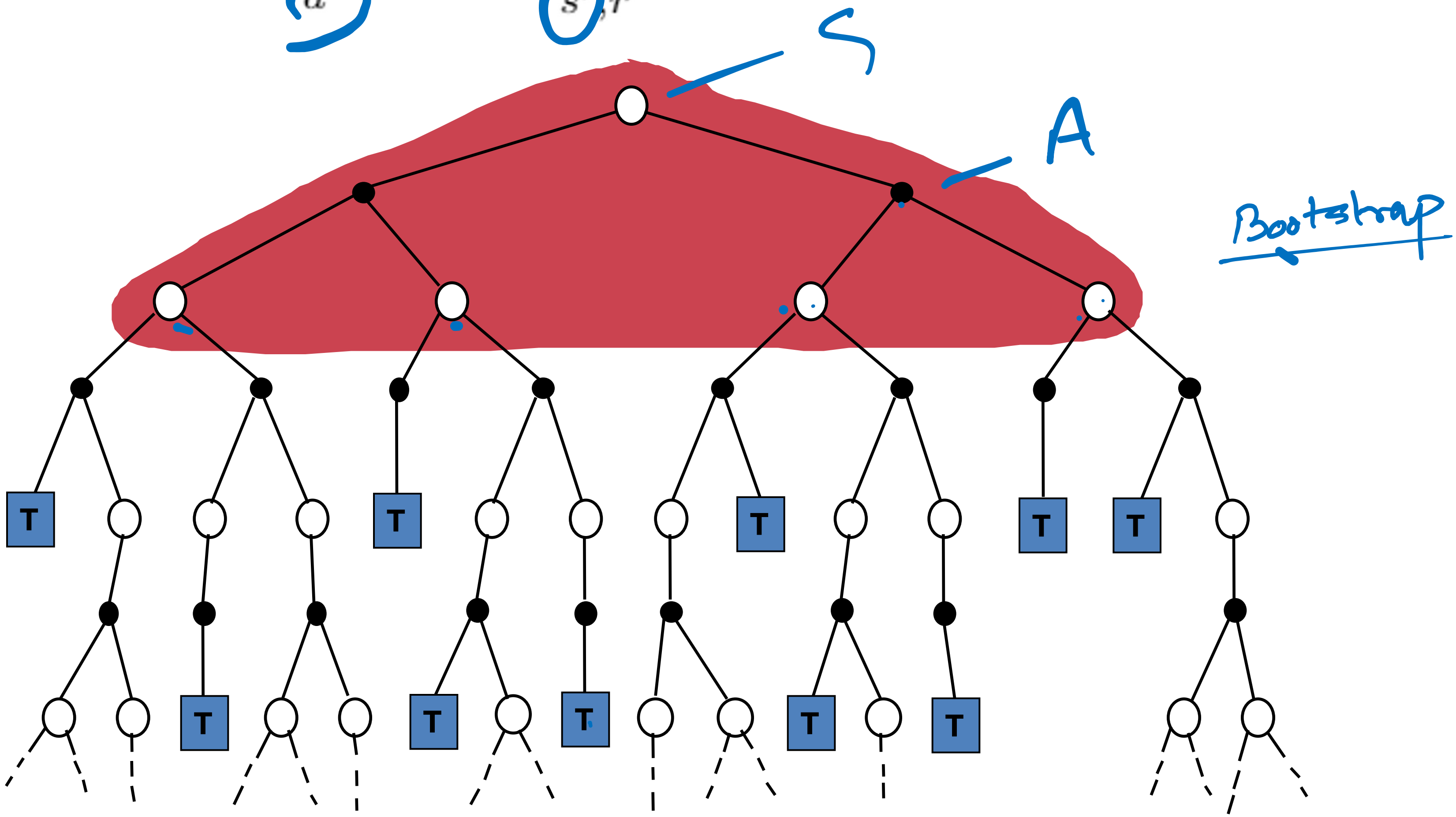
# Lecture 4:
# Temporal Difference Learning

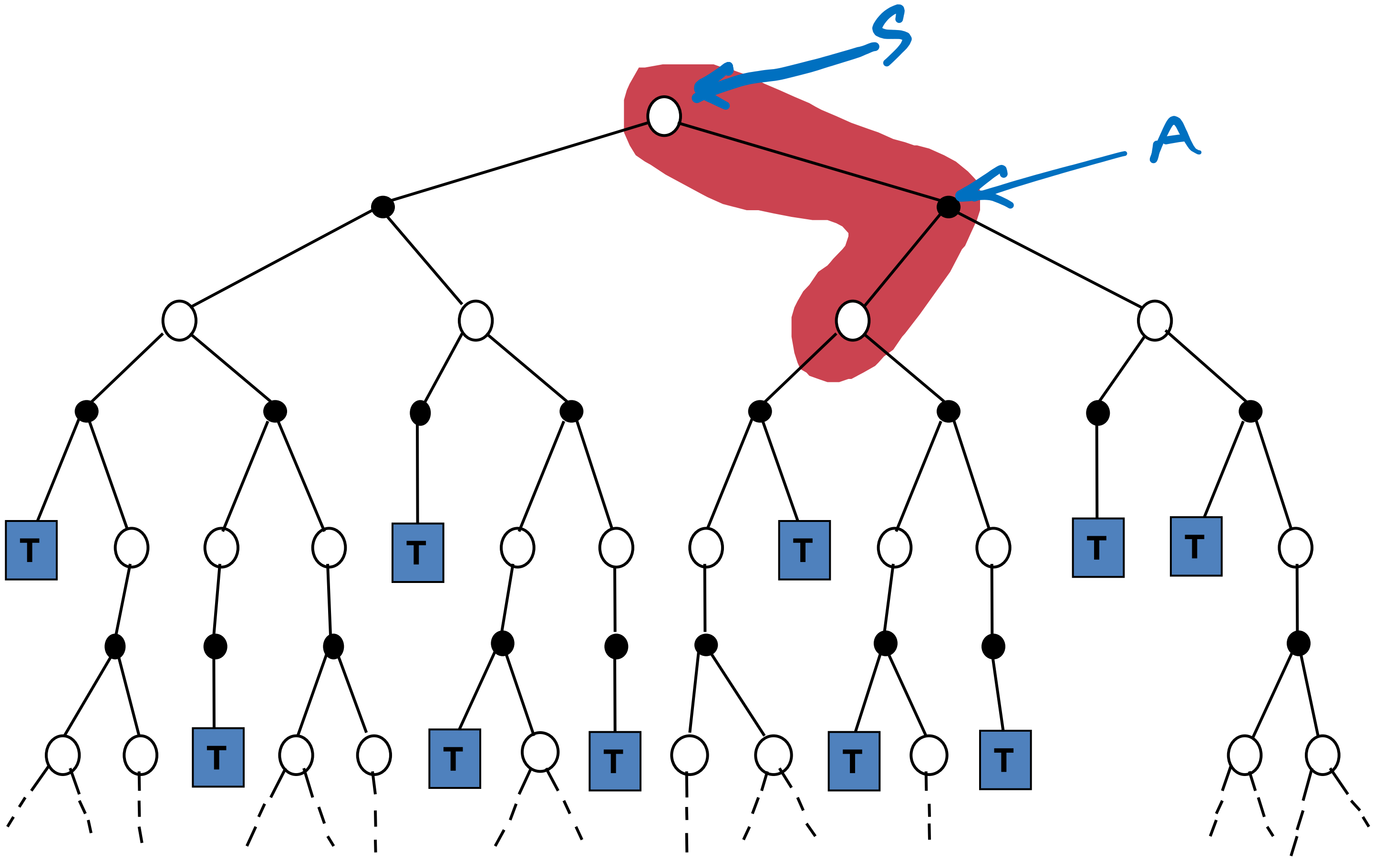
B. Ravindran

# Dynamic Programming

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

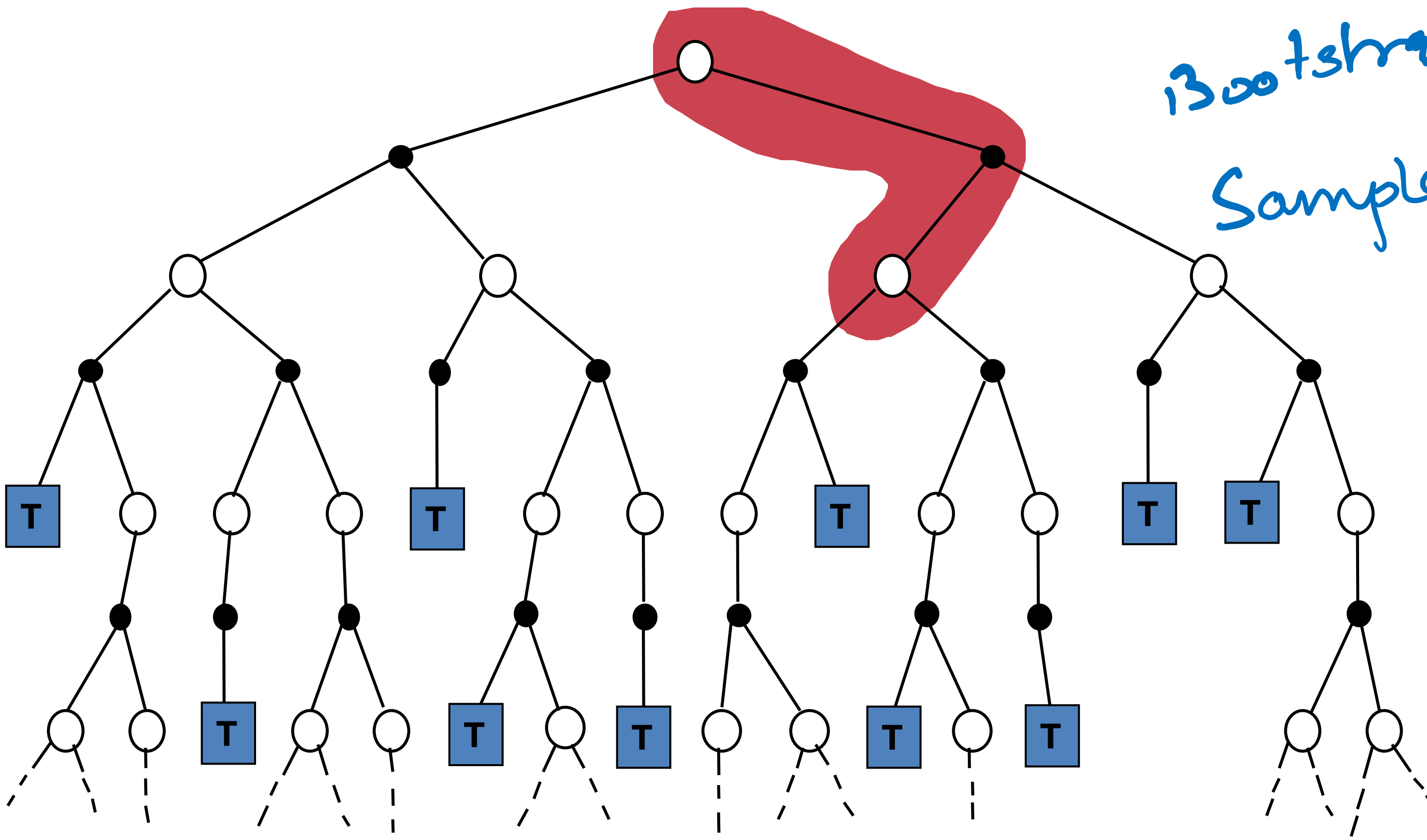


# Simplest "TD" Method



# Simplest "TD" Method

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma \underline{V(s_{t+1})} - V(s_t)\right]$$

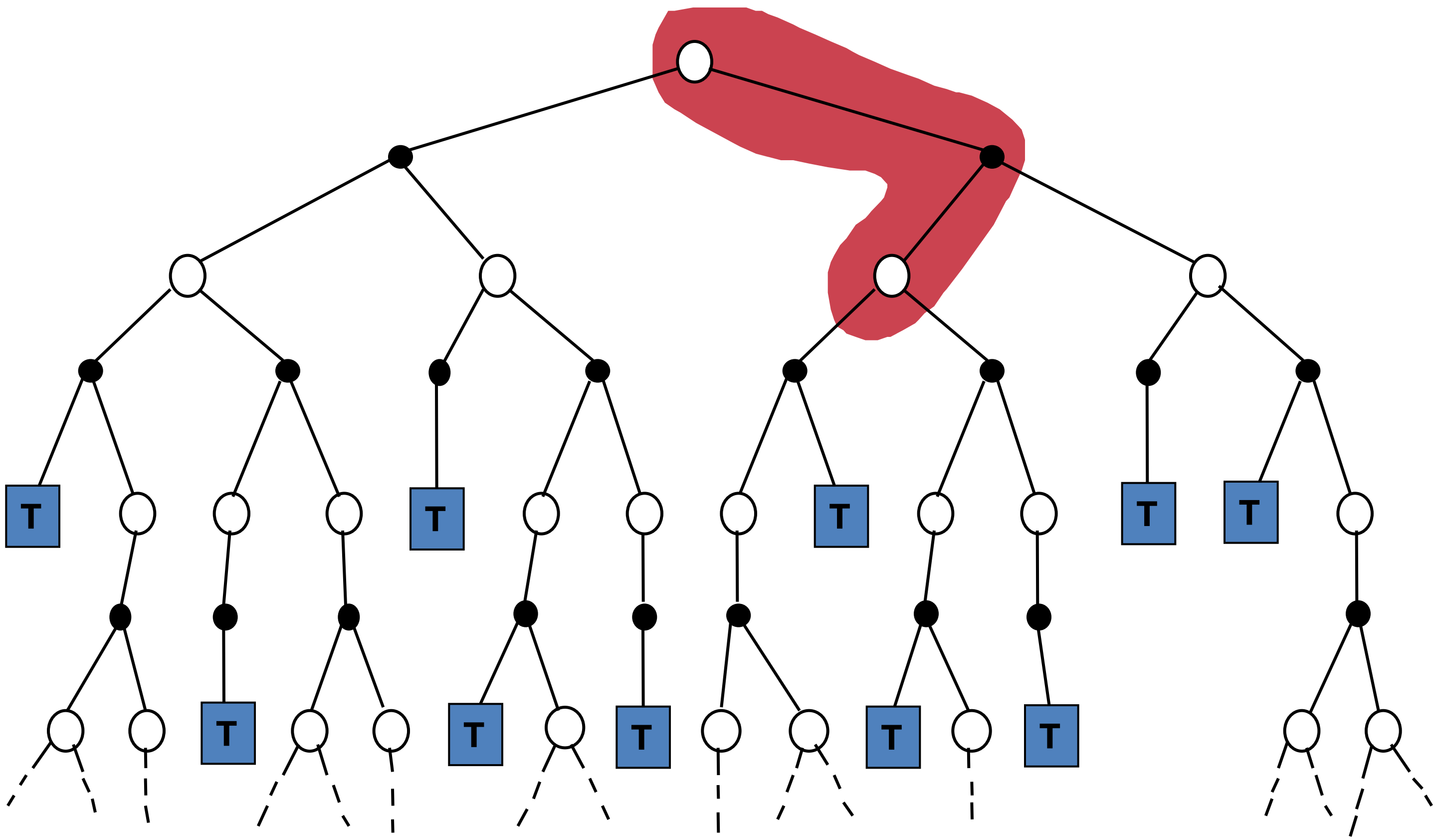



Bootstrap. ✓

Sample ✓

# Simplest "TD" Method

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



# Temporal Difference

- Simple rule to explain complex behaviors
- Intuition: Prediction of outcome at time $t+1$ is better than the prediction at time $t$. Hence use the later prediction to adjust the earlier prediction.
- Has had profound impact in behavioral psychology and neuroscience!

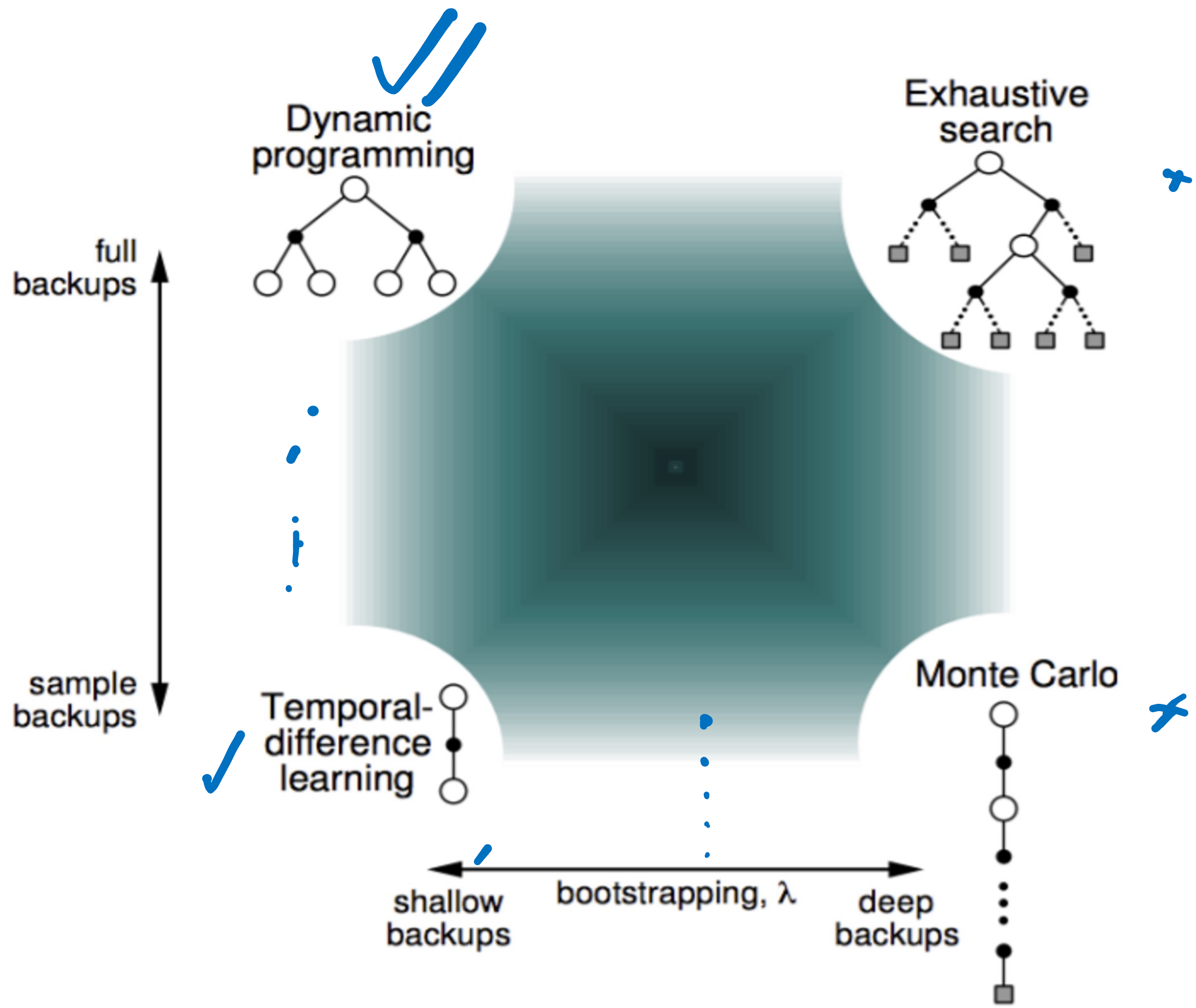
# Bootstrapping and Sampling

- Bootstrapping: Update using an estimate
  - DP and TD bootstrap
  - Monte Carlo does not bootstrap.

$$v_\pi(s) = E_\pi\{G_t \mid S_t = s\}$$

- Sampling: Update calculated using samples without model
  - TD and Monte Carlo sample.
  - DP (typically) does not sample

# Bootstrapping and Sampling



# Advantages of TD

$S_t$

$A_t$

$\sim S_{t+1}, R_{t+1}$

- TD methods do not require a model of the environment, only experience (sampling)

- TD methods can be fully incremental (bootstrapping)
  - You can learn before knowing the final outcome
    - Less memory
    - Less peak computation
  - You can learn without the final outcome
    - From incomplete sequences

Simulations

=

Sample Model.

$s', r \mid s,a \sim p(s', r \mid s,a)$

Full Model

Distributional model.

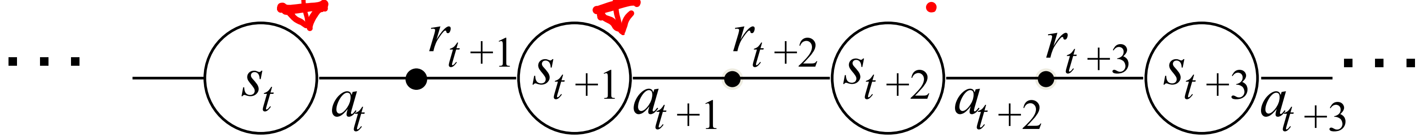
$p(s', r \mid s, a)$

# TD Prediction

- Policy Evaluation (the prediction problem): for a given policy, compute the state-value function. $\pi$ $v_\pi$
- No knowledge of $p$ and $r$, but access to the real system, or a "sample" model assumed.
- Uses "bootstrapping" and sampling

The simplest TD method, TD(0):

$$V_{k+1}(s_t) \leftarrow V_k(s_t) + \alpha[r_{t+1} + \gamma V_k(s_{t+1}) - V_k(s_t)]$$



# Stochastic Averaging Rule

$$E(x) \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

Let $\bar{x}_n$ be the average of the first $n$ samples

$$\begin{aligned}
\bar{x}_{n+1} &= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n\right) \\
&= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n + \bar{x}_n - \bar{x}_n\right) \\
&= \frac{1}{n+1}\left((n+1)\bar{x}_n + \left(x_{n+1} - \bar{x}_n\right)\right) \\
&= \bar{x}_n + \frac{1}{n+1}\left(x_{n+1} - \bar{x}_n\right) \\
&= \bar{x}_n + \alpha\left(x_{n+1} - \bar{x}_n\right)
\end{aligned}$$

new estimate = old estimate + $\alpha$ (new sample - old estimate)

# Stochastic Averaging Rule

$$E(x) \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

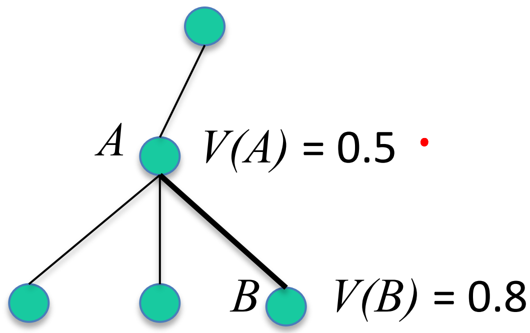
Let $\bar{x}_n$ be the average of the first $n$ samples

$$\begin{aligned}
\bar{x}_{n+1} &= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n\right) \\
&= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n + \bar{x}_n - \bar{x}_n\right) \\
&= \frac{1}{n+1}\left(\left(n+1\right)\bar{x}_n + \left(x_{n+1} - \bar{x}_n\right)\right) \\
&= \bar{x}_n + \frac{1}{n+1}\left(x_{n+1} - \bar{x}_n\right) \\
&= \bar{x}_n + \alpha\left(x_{n+1} - \bar{x}_n\right)
\end{aligned}$$

new estimate = old estimate + $\alpha$ (new sample - old estimate)

$$V_{k+1}(s_t) \leftarrow V_k(s_t) + \alpha[r_{t+1} + \gamma V_k(s_{t+1}) - V_k(s_t)]$$

# TD Update Example




Assuming :

reward, $r$, $\mathrm{A} \rightarrow \mathrm{B} : 0$

$\alpha : 0.2$

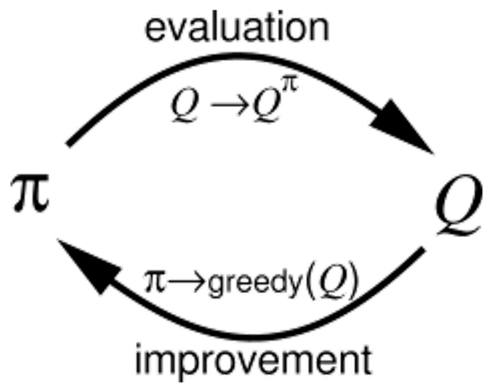
$\gamma : 0.9$

$$V(A) = V(A) + \alpha[r + \gamma V(B) - V(A)]$$

$$V(A) = 0.5 + 0.2[0 + 0.9 * 0.8 - 0.5] = 0.544$$

# TD Control

- The control problem: approximate optimal policies.
- Recall the idea of GPI:




- Policy evaluation: use TD(0) to evaluate value function.
- Policy improvement: make policy **greedy** wrt current value function.
- Note that we estimate action values rather than state values in the absence of a model.

# ε-Greedy Policies

$$a^* \leftarrow \arg\max_a Q(s,a)$$

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

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

- Any $\varepsilon$-greedy policy with respect to $Q$ following $\pi$ is an improvement over any $\varepsilon$-soft policy $\pi$ is assured by the policy improvement theorem

# Sarsa: On-Policy TD Control

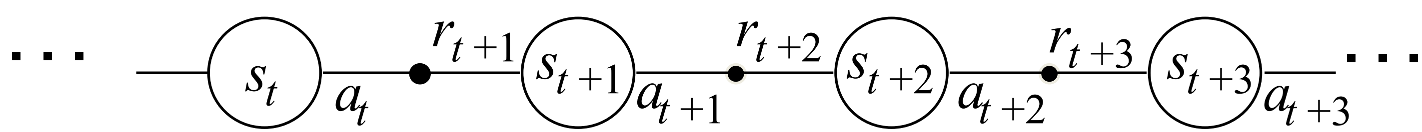
- In on-policy control, we try improving the policy used for making decisions.

**SARSA**

After every transition from a nonterminal state $s_t$, do this:

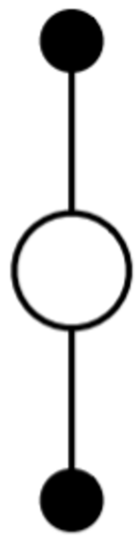
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

If $s_{t+1}$ is terminal, then $Q(s_{t+1}, a_{t+1}) = 0$.




- Convergence is guaranteed as long as
  - all state-action pairs are visited an infinite number of times
  - the policy converges in the limit to the greedy policy

# Sarsa: On-Policy TD Control



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

# Sarsa Algorithm

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode)

  Initialize $s$

  Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$ - greedy)

  Repeat (for each step of episode):

    Take action $a$, observe $r$, $s'$

    Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$ - greedy)

    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$

    $s \leftarrow s'; a \leftarrow a';$

  until $s$ is terminal

# Sarsa Algorithm

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode)

  Initialize $s$

  Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$ - greedy)

  Repeat (for each step of episode):

    Take action $a$, observe $r$, $s'$

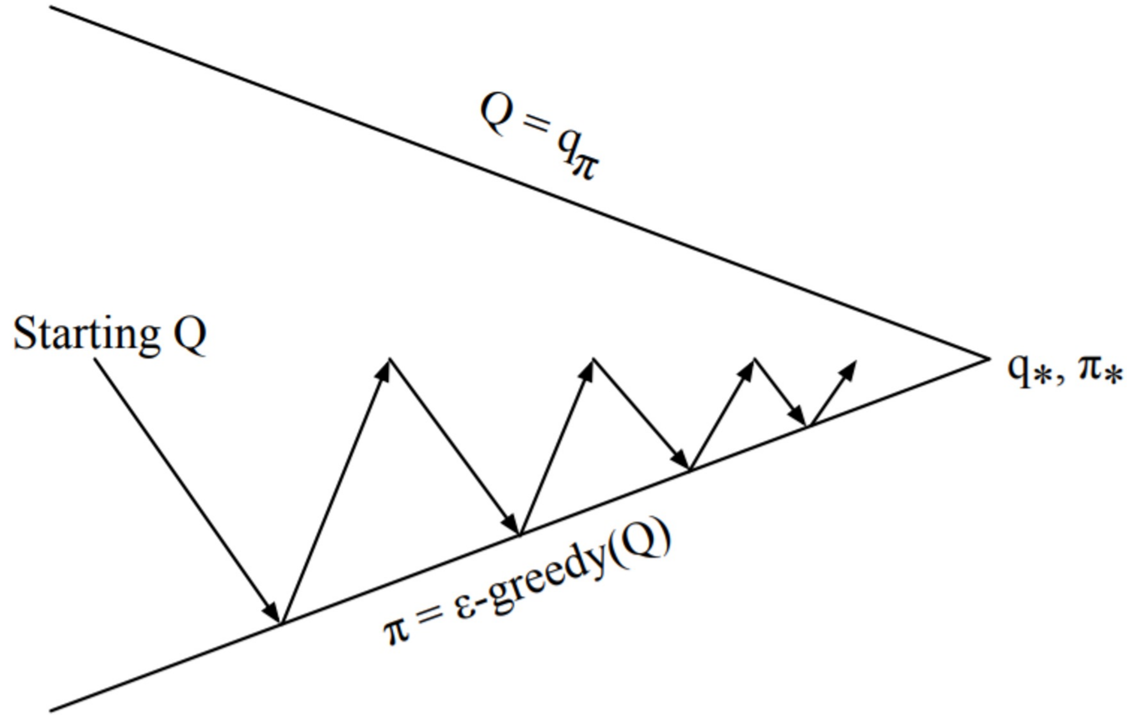
**IMPROVEMENT** Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$ - greedy)

**EVALUATION** $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$

    $s \leftarrow s'; a \leftarrow a';$

  until $s$ is terminal

# Sarsa Algorithm

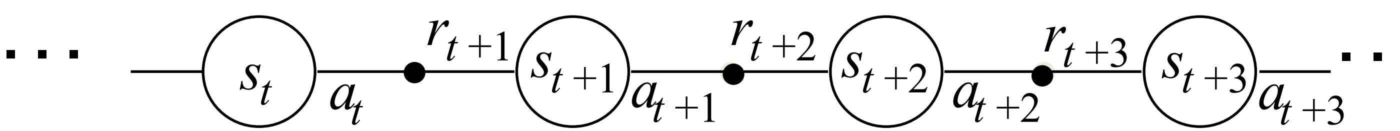



Every time-step:

Policy evaluation Sarsa, $Q \approx q_\pi$

Policy improvement $\epsilon$-greedy policy improvement

# Q-Learning

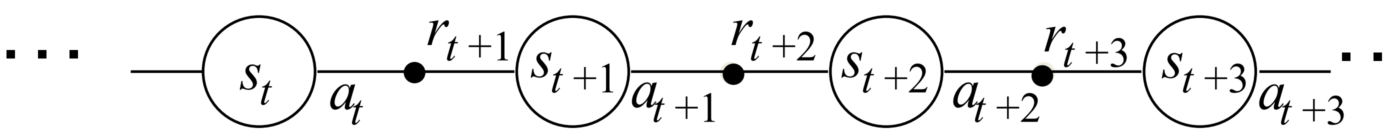



One-step Q-learning:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t)\right]$$

# Q-Learning

$$Q^*(s,a) = E\left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \,\middle|\, s_t = s, a_t = a \right\}$$




One-step Q-learning:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha\left[ r_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t) \right]$$

# Stochastic Averaging Rule

$$E(x) \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

Let $\overline{x}_n$ be the average of the first $n$ samples

$$\begin{aligned}
\overline{x}_{n+1} &= \frac{1}{n+1}\left(x_{n+1} + n\overline{x}_n\right) \\
&= \frac{1}{n+1}\left(x_{n+1} + n\overline{x}_n + \overline{x}_n - \overline{x}_n\right) \\
&= \frac{1}{n+1}\left(\left(n+1\right)\overline{x}_n + \left(x_{n+1} - \overline{x}_n\right)\right) \\
&= \overline{x}_n + \frac{1}{n+1}\left(x_{n+1} - \overline{x}_n\right) \\
&= \overline{x}_n + \alpha\left(x_{n+1} - \overline{x}_n\right)
\end{aligned}$$

new estimate = old estimate + $\alpha$ (new sample - old estimate)

# Stochastic Averaging Rule

$$E(x) \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

Let $\bar{x}_n$ be the average of the first $n$ samples

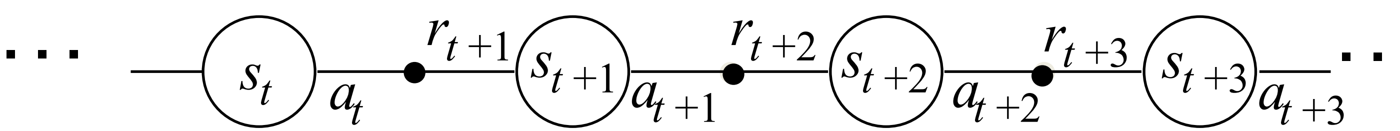
$$\begin{aligned}
\bar{x}_{n+1} &= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n\right) \\
&= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n + \bar{x}_n - \bar{x}_n\right) \\
&= \frac{1}{n+1}\left(\left(n+1\right)\bar{x}_n + \left(x_{n+1} - \bar{x}_n\right)\right) \\
&= \bar{x}_n + \frac{1}{n+1}\left(x_{n+1} - \bar{x}_n\right) \\
&= \bar{x}_n + \alpha\left(x_{n+1} - \bar{x}_n\right)
\end{aligned}$$

new estimate = old estimate + $\alpha$ (new sample - old estimate)

$$\hat{q}\left(s_t, a_t\right) \leftarrow \hat{q}\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma \max_a \hat{q}\left(s_{t+1}, a\right) - \hat{q}\left(s_t, a_t\right)\right]$$

# Q-Learning

$$Q^*(s,a) = E\left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \,\middle|\, s_t = s, a_t = a \right\}$$

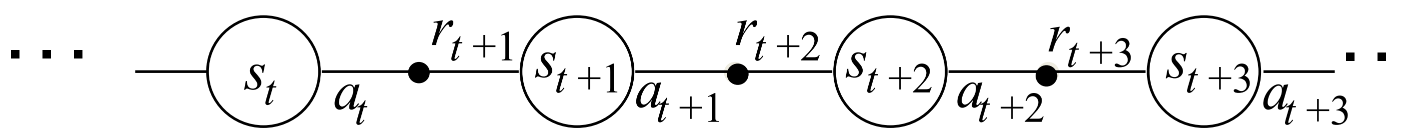



One-step Q-learning:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_{a} \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t) \right]$$

# Q-Learning

$$Q^*(s,a) = E\left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1},a') \,\middle|\, s_t = s, a_t = a \right\}$$




One-step Q-learning:

Temporal Difference

$$\hat{q}(s_t,a_t) \leftarrow \hat{q}(s_t,a_t) + \alpha\left[ r_{t+1} + \gamma \max_{a} \hat{q}(s_{t+1},a) - \hat{q}(s_t,a_t) \right]$$

# Q-Learning: Off-Policy TD Control

- In off-policy control, we have two policies:
  - the behavior policy – used to generate behavior
  - estimation policy – the policy that is being evaluated and improved.

**Q-learning**

One-step Q-learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

# Q-learning Algorithm

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode) :

  Initialize $s$

  Repeat (for each step of episode) :

    Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$ - greedy)

    Take action $a$, observev $r$, $s'$

    $Q(s,a) \leftarrow Q(s,a) \ + \ \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

    $s \leftarrow s'$;

until $s$ is terminal

# Q-learning Algorithm

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode) :

  Initialize $s$

  Repeat (for each step of episode) :

    Choose $a$ from $s$ using policy derived from $Q$ (~~e.g., $\varepsilon$ - greedy~~)

    Take action $a$, observev $r$, $s'$

    $Q(s,a) \leftarrow Q(s,a) \; + \; \alpha\left[r + \gamma \max_{a'} Q(s',a')\text{-}Q(s,a)\right]$

    $s \leftarrow s'$;

until $s$ is terminal

# Q-learning Algorithm

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode) :

  Initialize $s$

  Repeat (for each step of episode) :

    Choose $a$ from $s$ ~~using policy derived from $Q$ (e.g., $\varepsilon$ - greedy)~~

    Take action $a$, observev $r$, $s'$

    $Q(s,a) \leftarrow Q(s,a) \ + \ \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

    $s \leftarrow s'$;

until $s$ is terminal

# Q-learning Algorithm

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

# Cliff Walking: SARSA vs Q-learning