# Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
- MC learns from *complete* episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to *episodic* MDPs
  - All episodes must terminate

# Monte-Carlo Policy Evaluation

- Goal: learn $v_\pi$ from episodes of experience under policy $\pi$

$$S_1, A_1, R_2, ..., S_k \sim \pi$$

- Recall that the *return* is the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$$

- Recall that the value function is the expected return:

$$v_\pi(s) = \mathbb{E}_\pi \left[ G_t \mid S_t = s \right]$$

- Monte-Carlo policy evaluation uses *empirical mean* return instead of *expected* return

# First-Visit Monte-Carlo Policy Evaluation

- To evaluate state $s$
- The first time-step $t$ that state $s$ is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

# Every-Visit Monte-Carlo Policy Evaluation

- To evaluate state $s$
- Every time-step $t$ that state $s$ is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

# Incremental Mean

The mean $\mu_1, \mu_2, ...$ of a sequence $x_1, x_2, ...$ can be computed incrementally,

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}
$$

# Incremental Monte-Carlo Updates

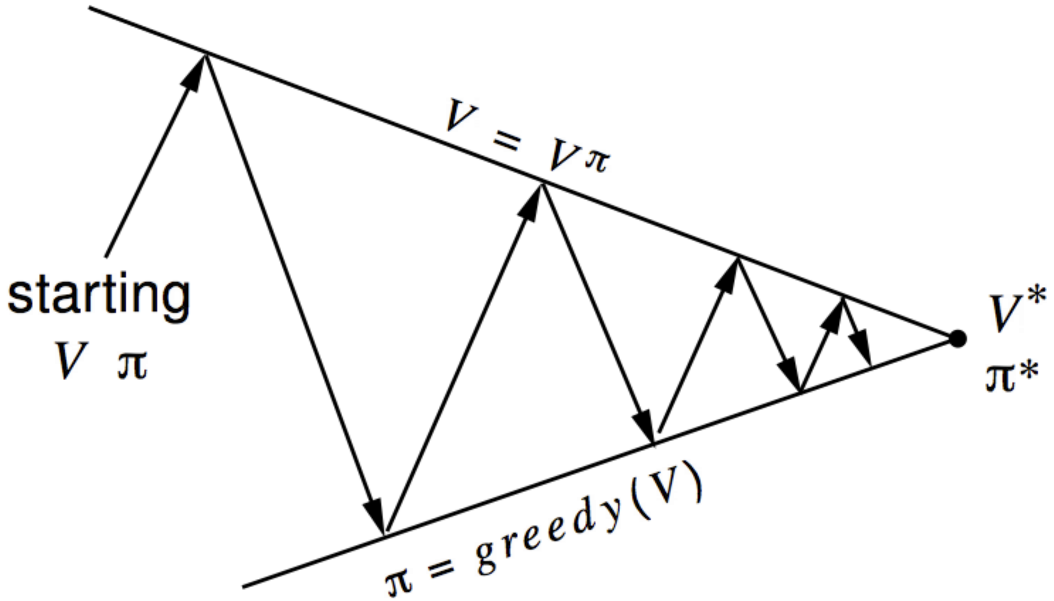
- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, ..., S_T$
- For each state $S_t$ with return $G_t$

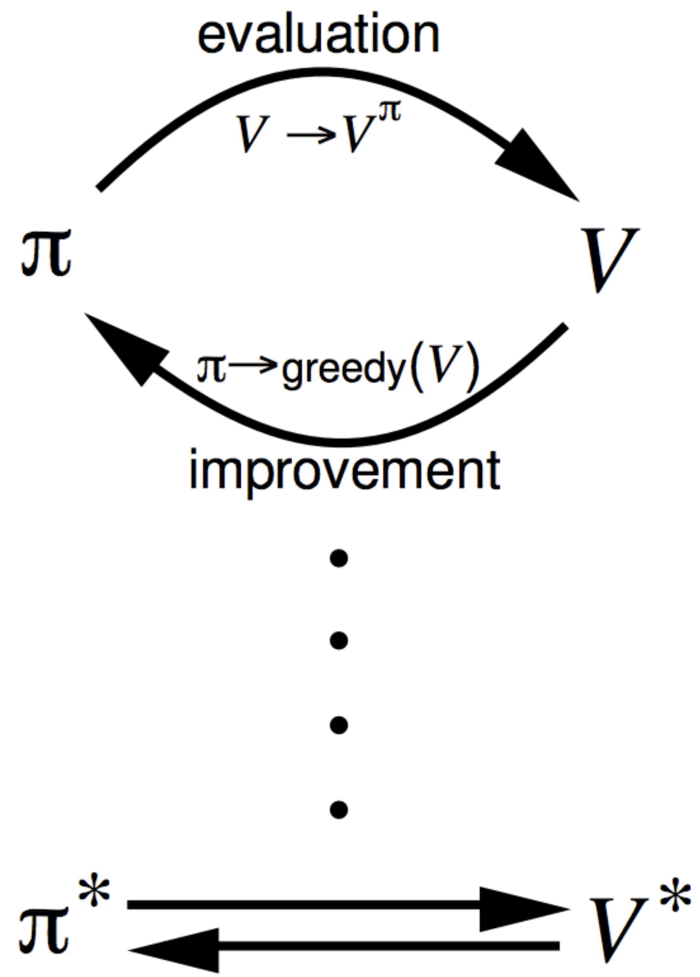
$$N(S_t) \leftarrow N(S_t) + 1$$
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left(G_t - V(S_t)\right)$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

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

# Generalised Policy Iteration (Refresher)

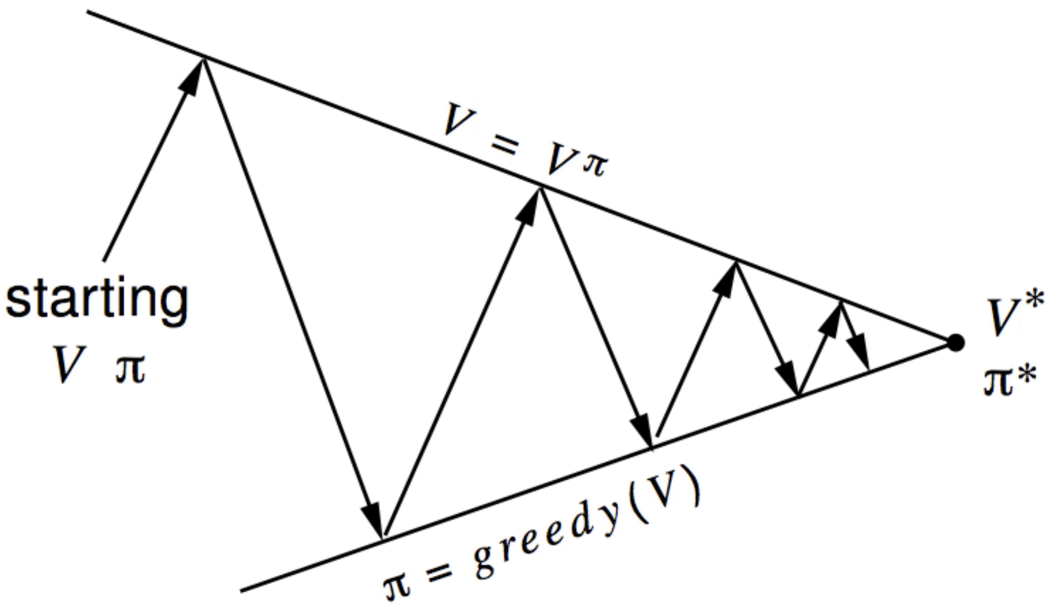



Policy evaluation Estimate $v_\pi$
e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
e.g. Greedy policy improvement

# Generalised Policy Iteration With Monte-Carlo Evaluation




Policy evaluation Monte-Carlo policy evaluation, $V = v_\pi$?

Policy improvement Greedy policy improvement?

# Model-Free Policy Iteration Using Action-Value Function

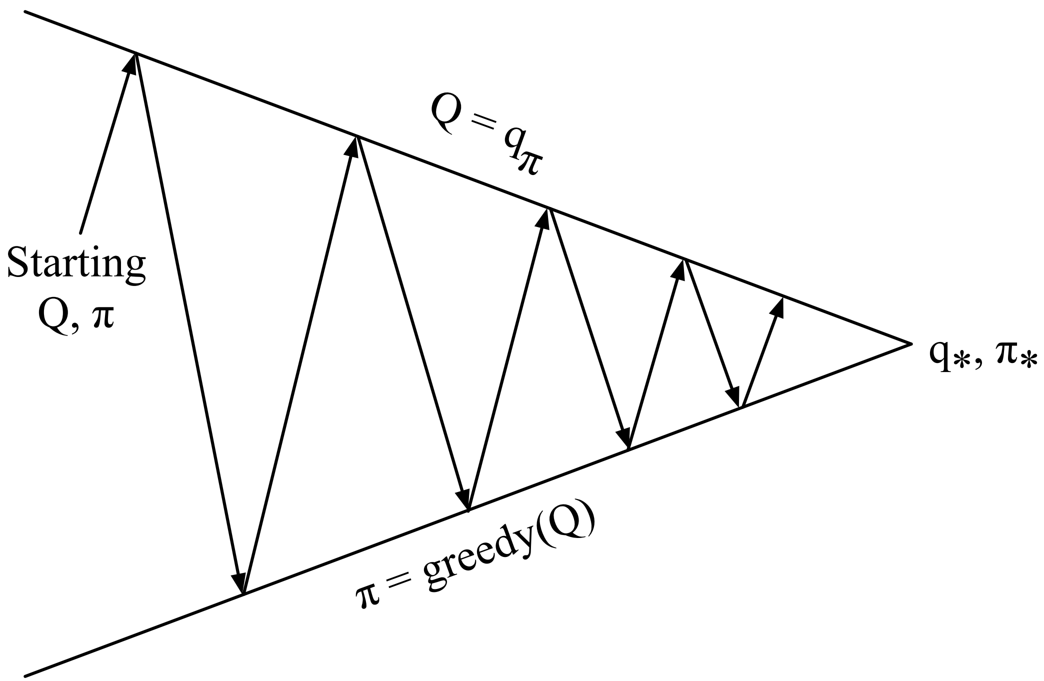
- Greedy policy improvement over $V(s)$ requires model of MDP

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

- Greedy policy improvement over $Q(s, a)$ is model-free

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(s, a)$$

# Generalised Policy Iteration with Action-Value Function




**Policy evaluation** Monte-Carlo policy evaluation, $Q = q_\pi$

**Policy improvement** Greedy policy improvement?

# Example of Greedy Action Selection

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

- There are two doors in front of you.
- You open the left door and get reward 0
  $V(left) = 0$
- You open the right door and get reward +1
  $V(right) = +1$
- You open the right door and get reward +3
  $V(right) = +2$
- You open the right door and get reward +2
  $V(right) = +2$

$$\vdots$$

- Are you sure you've chosen the best door?

# $\epsilon$-Greedy Exploration

- Simplest idea for ensuring continual exploration
- All $m$ actions are tried with non-zero probability
- With probability $1 - \epsilon$ choose the greedy action
- With probability $\epsilon$ choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

# $\epsilon$-Greedy Policy Improvement

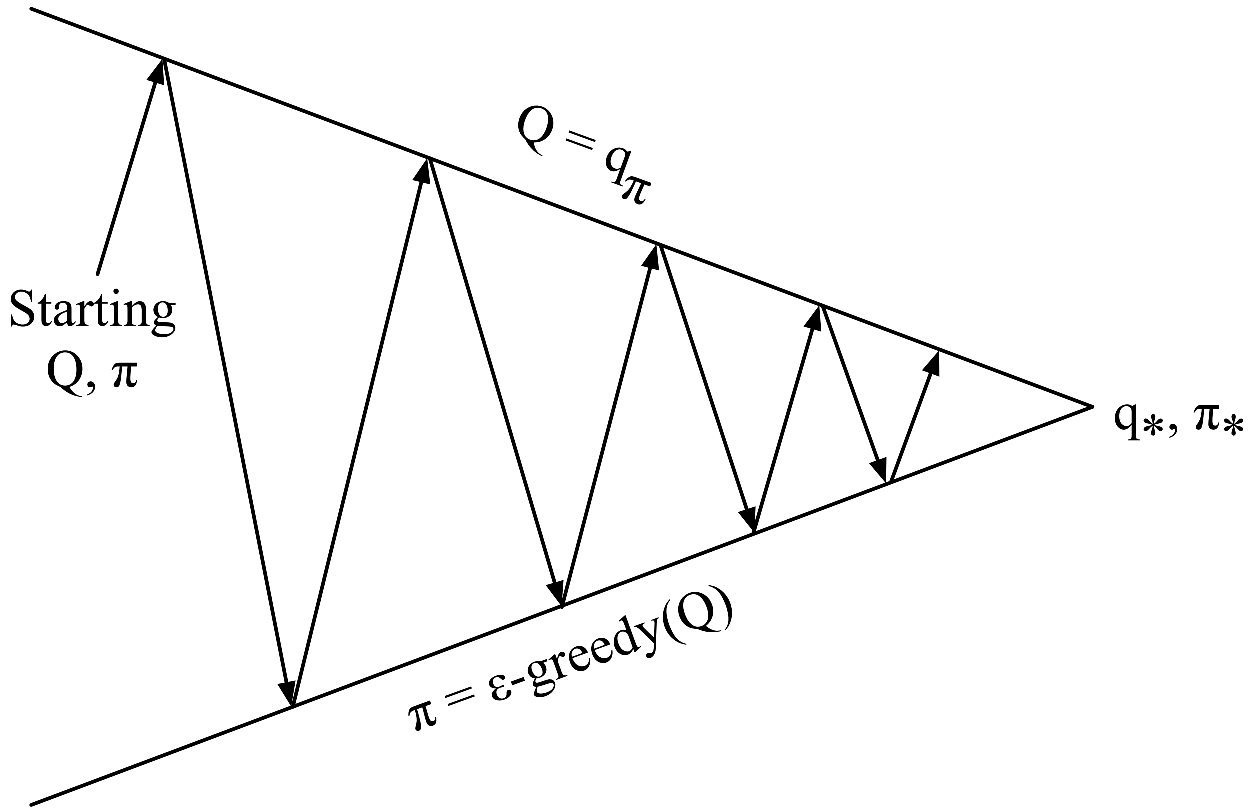
**Theorem**

*For any $\epsilon$-greedy policy $\pi$, the $\epsilon$-greedy policy $\pi'$ with respect to $q_\pi$ is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$*

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\
&= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} q_\pi(s, a) \\
&\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1 - \epsilon} q_\pi(s, a) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s)
\end{aligned}
$$

Therefore from policy improvement theorem, $v_{\pi'}(s) \geq v_\pi(s)$

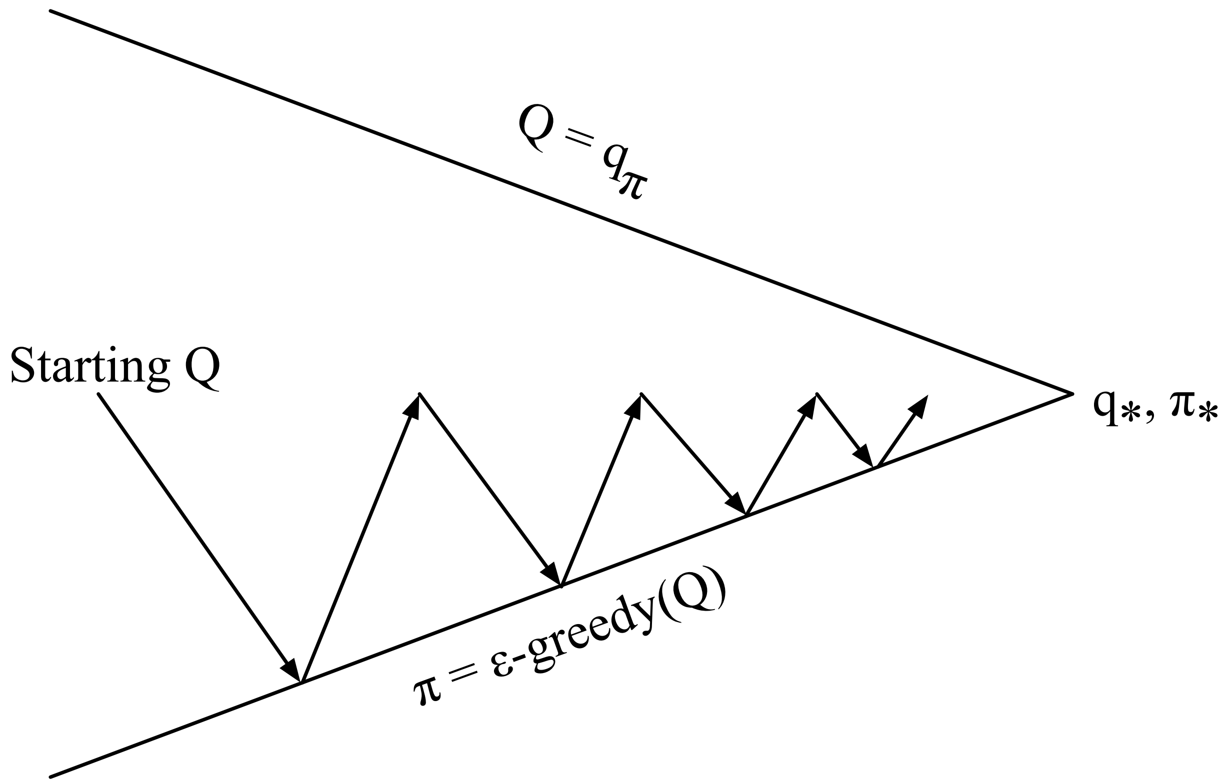
# Monte-Carlo Policy Iteration




**Policy evaluation** Monte-Carlo policy evaluation, $Q = q_\pi$

**Policy improvement** $\epsilon$-greedy policy improvement

# Monte-Carlo Control




Every episode:

**Policy evaluation** Monte-Carlo policy evaluation, $Q \approx q_\pi$

**Policy improvement** $\epsilon$-greedy policy improvement

# GLIE

## Definition

*Greedy in the Limit with Infinite Exploration* (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \to \infty} \pi_k(a|s) = \mathbf{1}(a = \operatorname*{argmax}_{a' \in \mathcal{A}} Q_k(s, a'))$$

- For example, $\epsilon$-greedy is GLIE if $\epsilon$ reduces to zero at $\epsilon_k = \frac{1}{k}$

# GLIE Monte-Carlo Control

- Sample $k$th episode using $\pi$: $\{S_1, A_1, R_2, ..., S_T\} \sim \pi$
- For each state $S_t$ and action $A_t$ in the episode,

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \left(G_t - Q(S_t, A_t)\right)$$

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$
$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

**Theorem**

*GLIE Monte-Carlo control converges to the optimal action-value function,* $Q(s, a) \rightarrow q_*(s, a)$

# Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

# MC and TD

- Goal: learn $v_\pi$ online from experience under policy $\pi$
- 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* return $R_{t+1} + \gamma V(S_{t+1})$

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

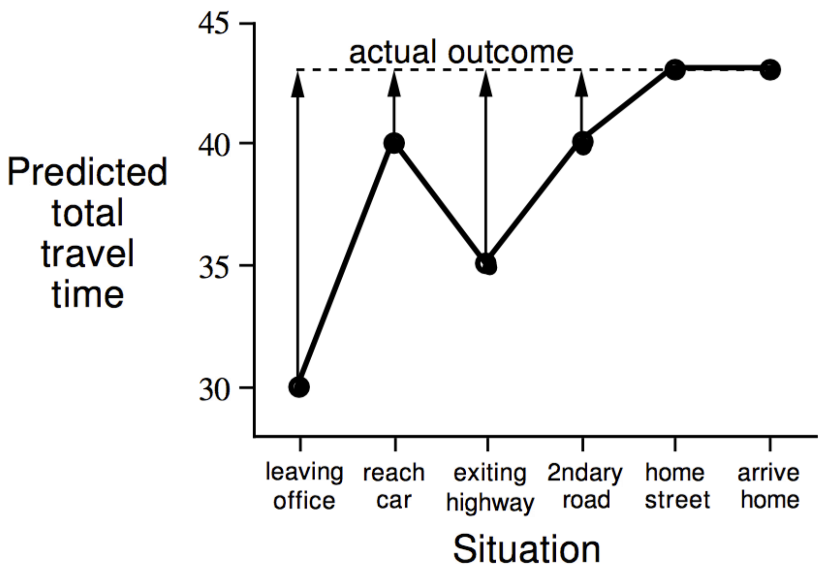
  - $R_{t+1} + \gamma V(S_{t+1})$ is called the *TD target*
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

# Driving Home Example

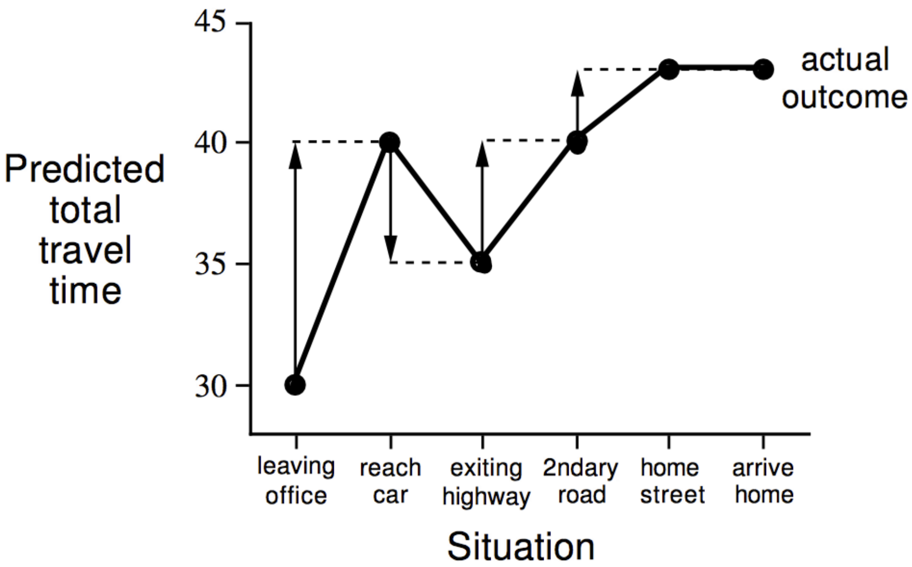
| **State** | **Elapsed Time (minutes)** | **Predicted Time to Go** | **Predicted Total Time** |
|---|---|---|---|
| leaving office | 0 | 30 | 30 |
| reach car, raining | 5 | 35 | 40 |
| exit highway | 20 | 15 | 35 |
| behind truck | 30 | 10 | 40 |
| home street | 40 | 3 | 43 |
| arrive home | 43 | 0 | 43 |

# Driving Home Example: MC vs. TD

Changes recommended by Monte Carlo methods ($\alpha$=1)




Changes recommended by TD methods ($\alpha$=1)



# Advantages and Disadvantages of MC vs. 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

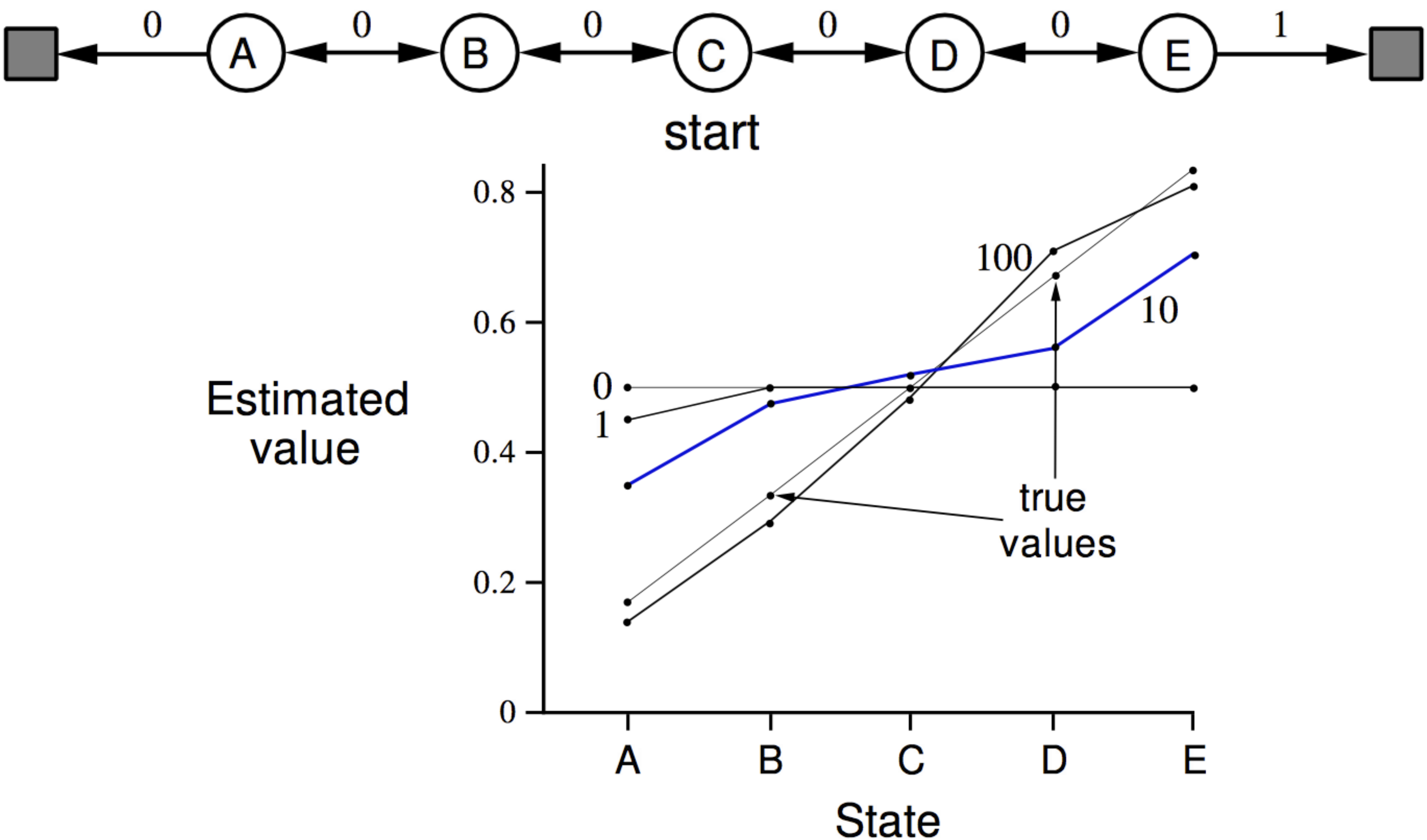
# Bias/Variance Trade-Off

- 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

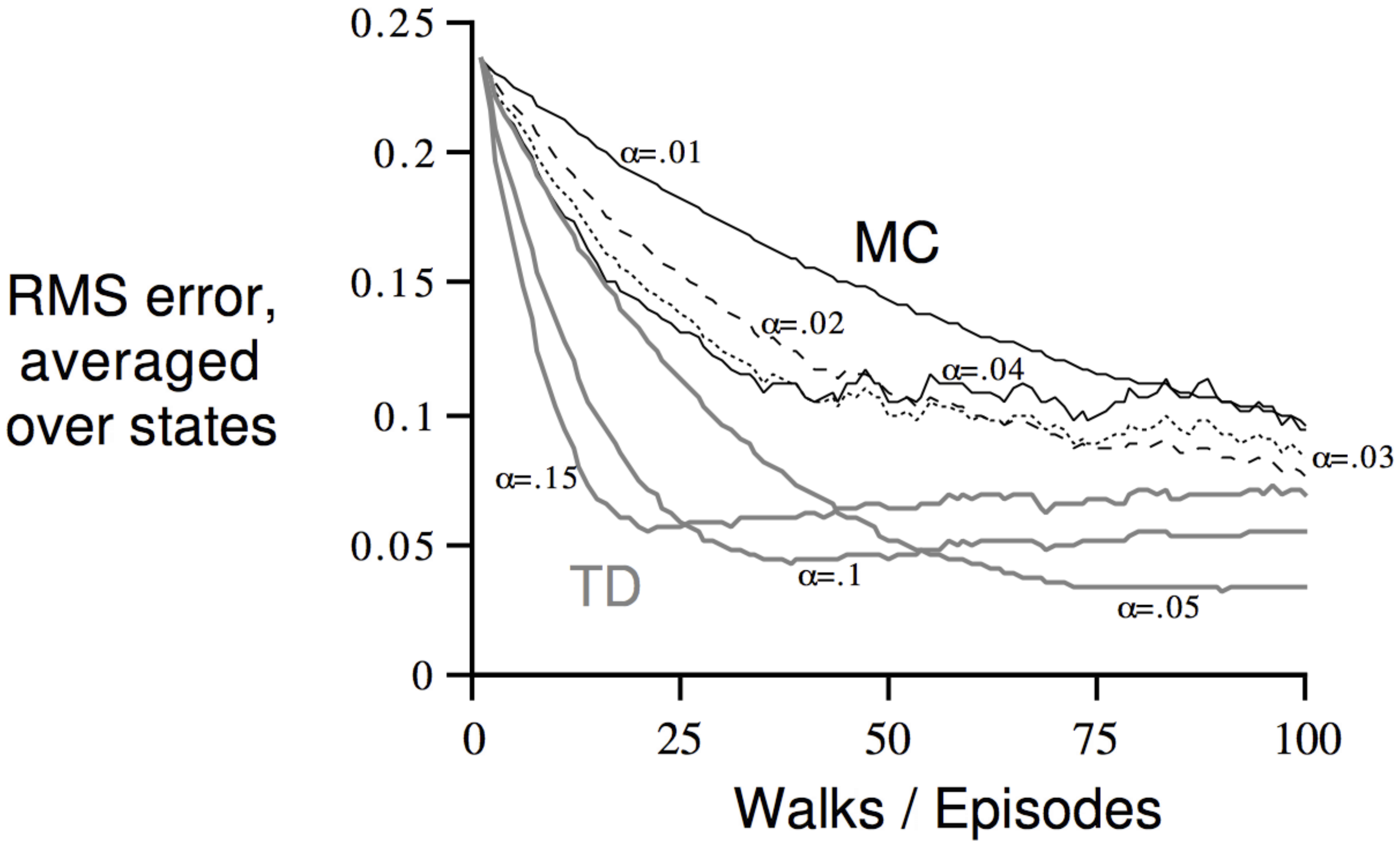
# Advantages and Disadvantages of MC vs. TD (2)

- 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

# Random Walk Example



# Random Walk: MC vs. TD



# Batch MC and TD

- MC and TD converge: $V(s) \rightarrow v_\pi(s)$ as experience $\rightarrow \infty$
- But what about batch solution for finite experience?

$$s_1^1, a_1^1, r_2^1, ..., s_{T_1}^1$$
$$\vdots$$
$$s_1^K, a_1^K, r_2^K, ..., s_{T_K}^K$$

  - e.g. Repeatedly sample episode $k \in [1, K]$
  - Apply MC or TD(0) to episode $k$

# AB Example

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

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

$\mathrm{B}, 1$

$\mathrm{B}, 1$

$\mathrm{B}, 1$

$\mathrm{B}, 1$

$\mathrm{B}, 1$

$\mathrm{B}, 1$

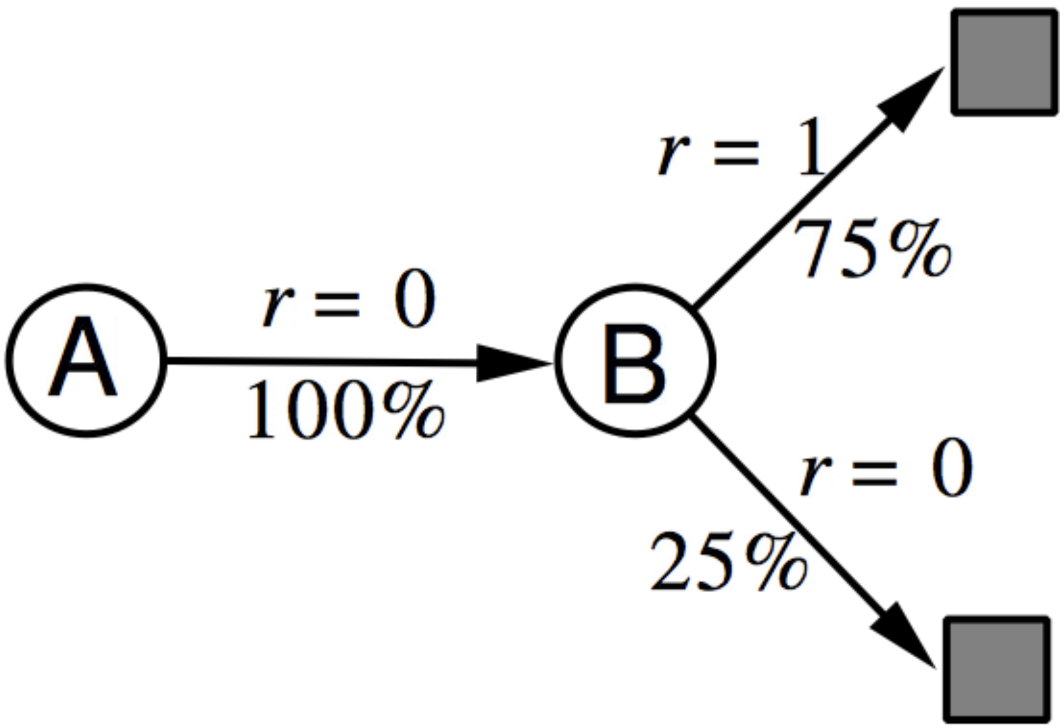
$\mathrm{B}, 0$

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

# AB 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 $V(A), V(B)$?

# Certainty Equivalence

- MC converges to solution with minimum mean-squared error
  - Best fit to the observed returns

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}\left(G_t^k - V(s_t^k)\right)^2$$

  - In the AB example, $V(A) = 0$
- TD(0) converges to solution of max likelihood Markov model
  - Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ that best fits the data

$$\hat{\mathcal{P}}_{s,s'}^{a} = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{\mathcal{R}}_{s}^{a} = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$$

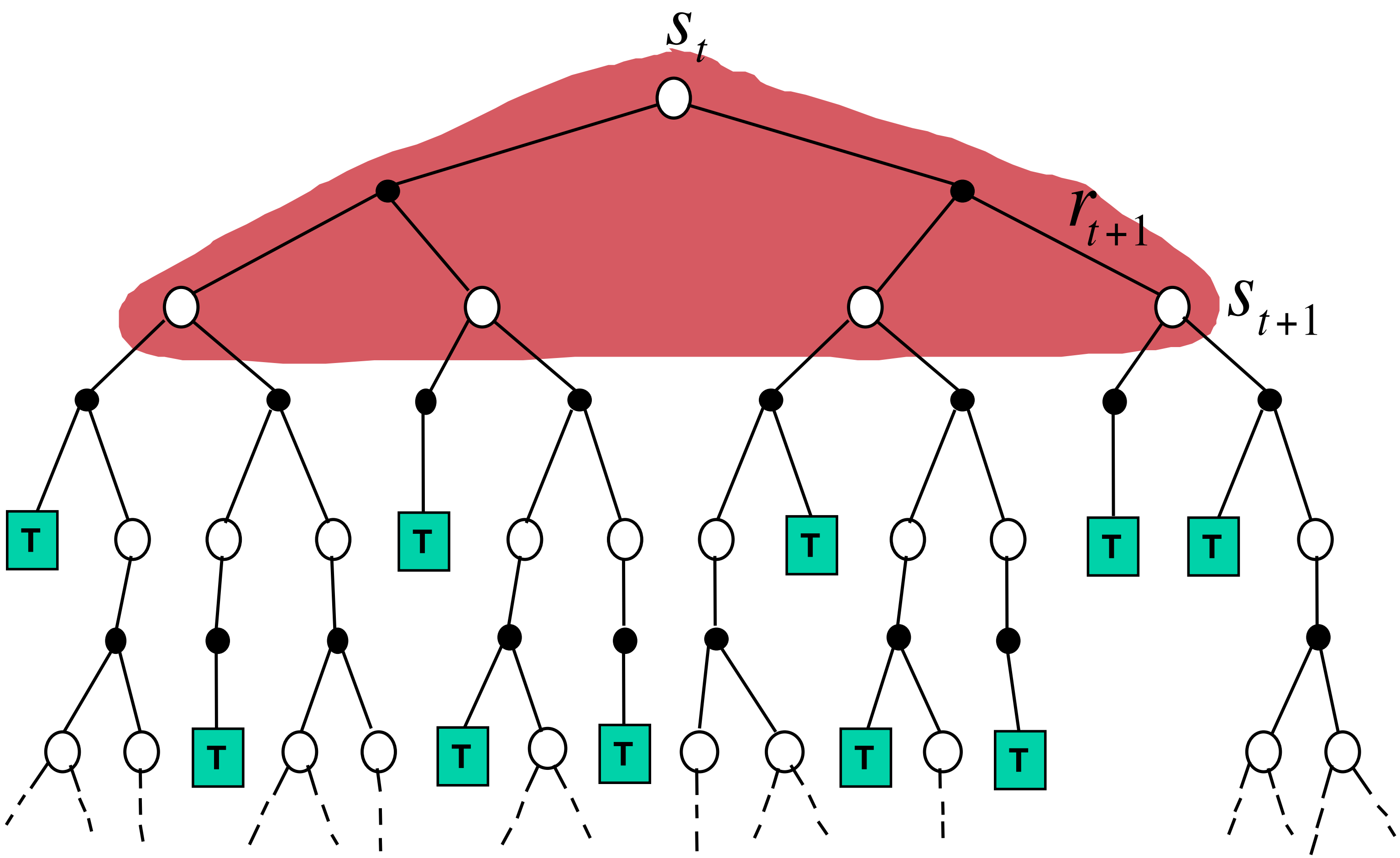
  - In the AB example, $V(A) = 0.75$

# Advantages and Disadvantages of MC vs. TD (3)

- TD exploits Markov property
  - Usually more efficient in Markov environments
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments

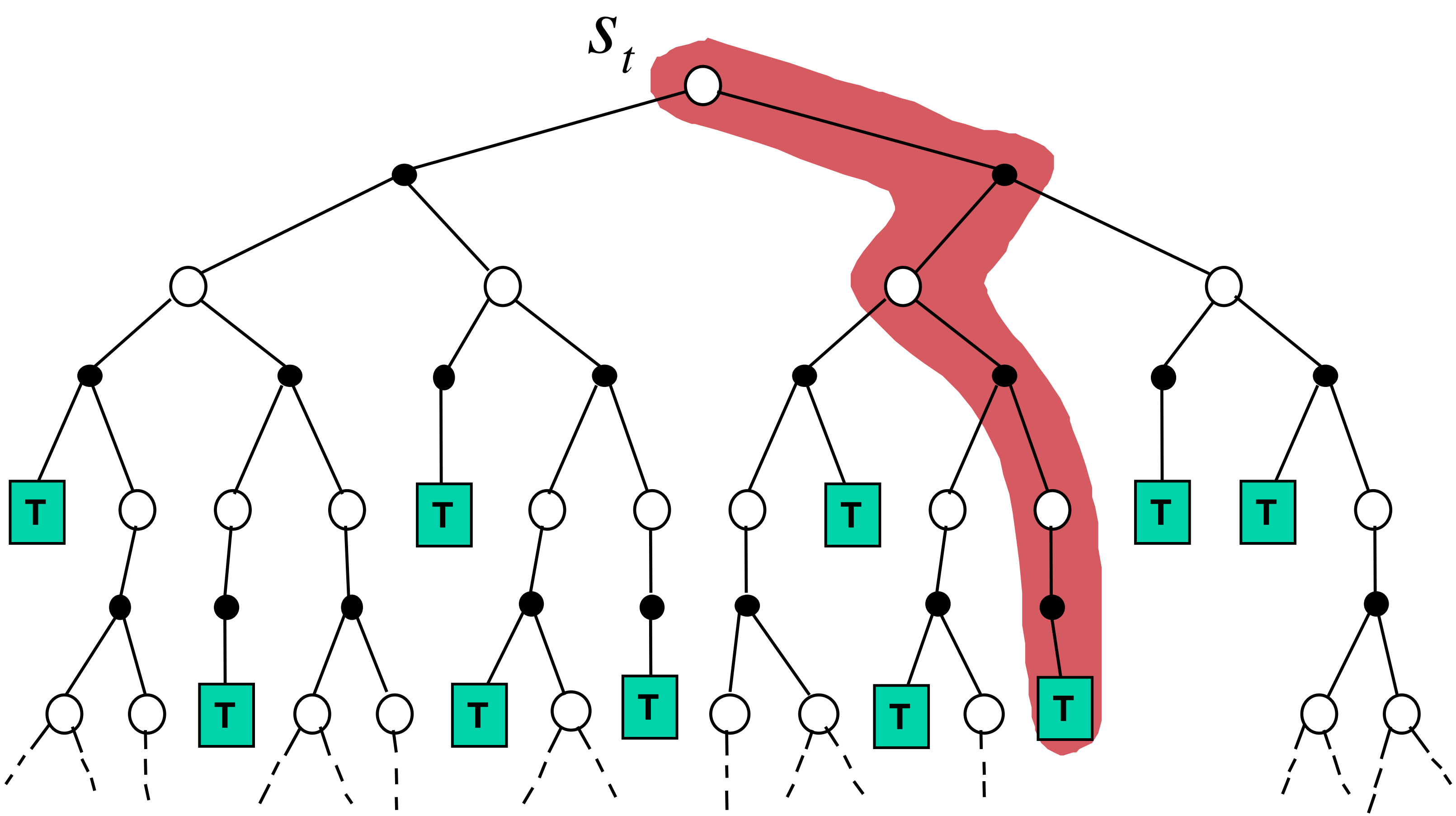
# Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



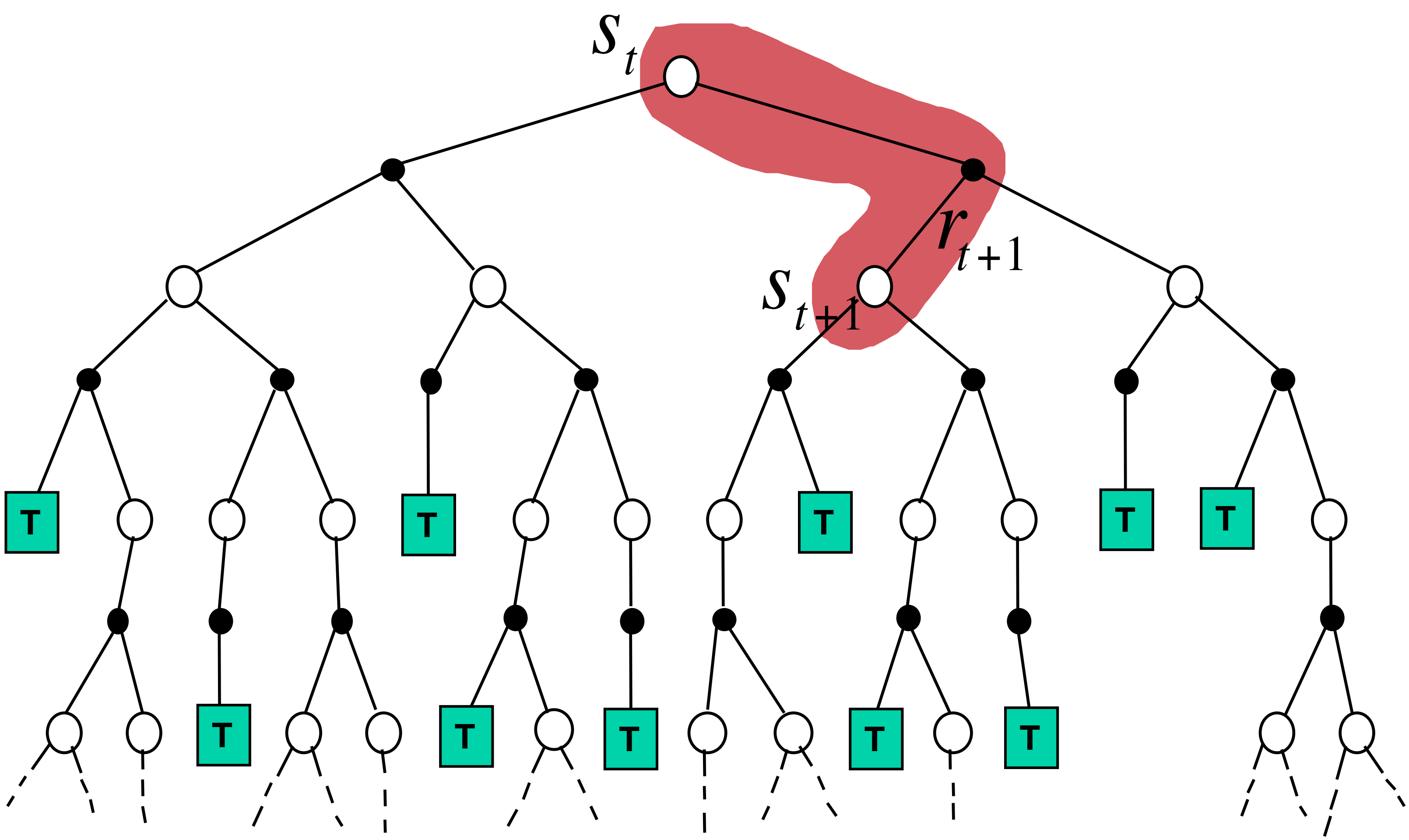
# Monte-Carlo Backup

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



# Temporal-Difference Backup

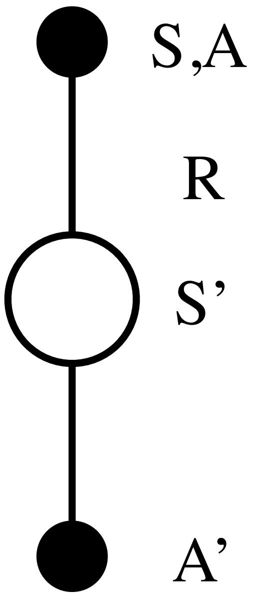
$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



# MC vs. TD Control

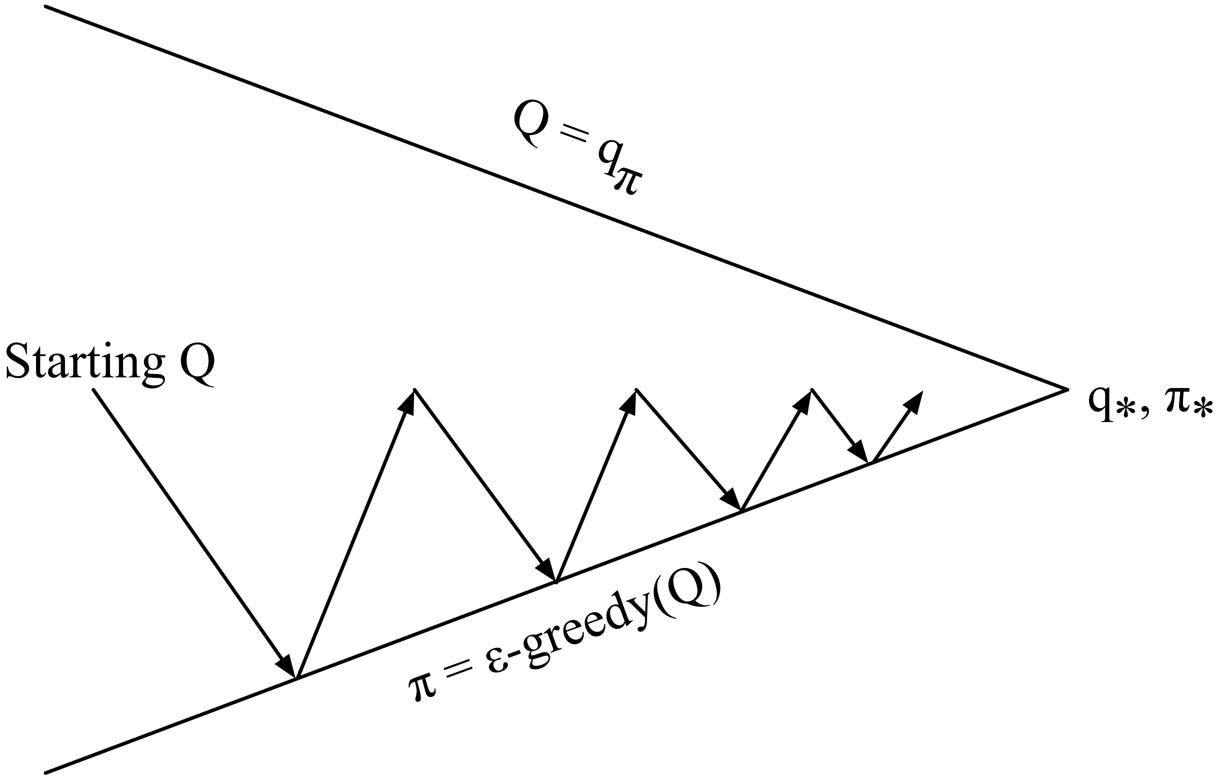
- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to $Q(S, A)$
  - Use $\epsilon$-greedy policy improvement
  - Update every time-step

# Updating Action-Value Functions with Sarsa




$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma Q(S', A') - Q(S, A)\right)$$

# On-Policy Control With Sarsa




Every time-step:

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

Policy improvement $\epsilon$-greedy policy improvement

# Sarsa Algorithm for On-Policy Control

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
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)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until $S$ is terminal

# Convergence of Sarsa

## Theorem

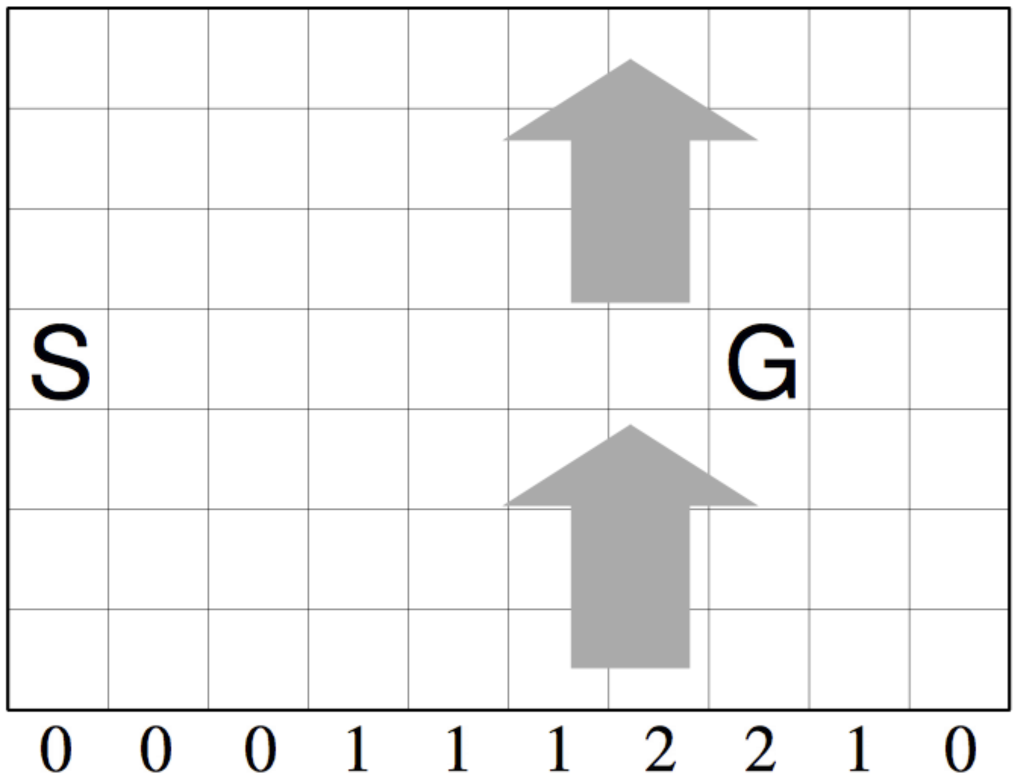
*Sarsa converges to the optimal action-value function, $Q(s, a) \to q_*(s, a)$, under the following conditions:*

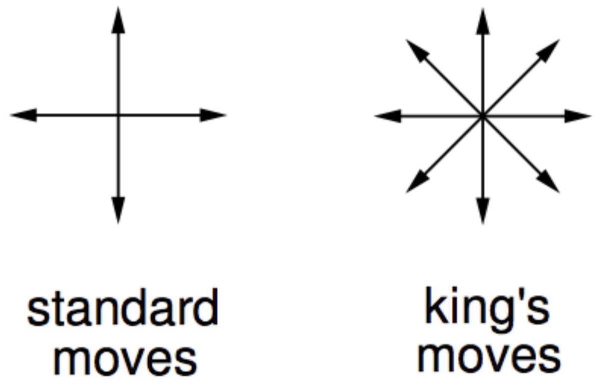
- *GLIE sequence of policies $\pi_t(a|s)$*
- *Robbins-Monro sequence of step-sizes $\alpha_t$*

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

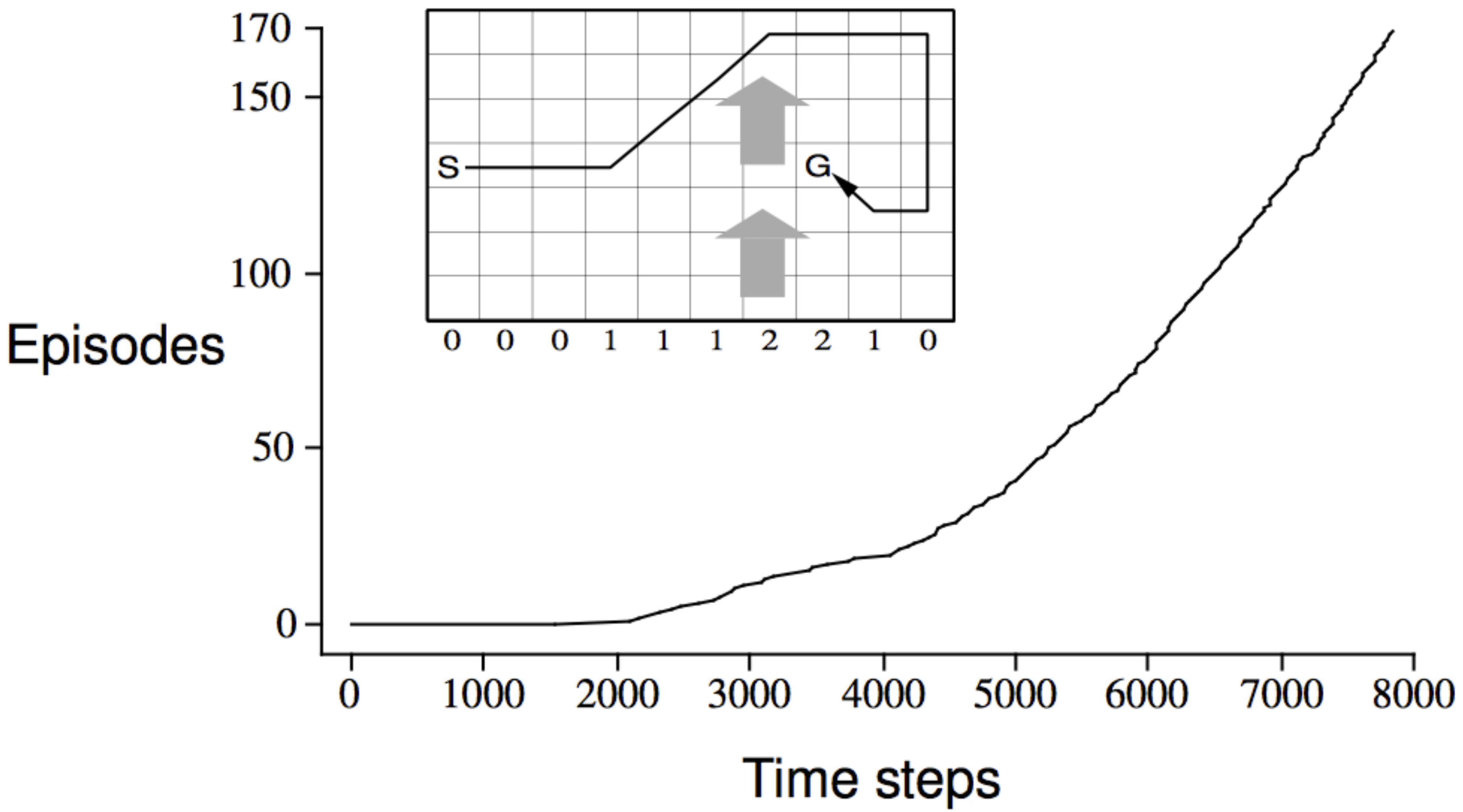
# Windy Gridworld Example




- Reward = -1 per time-step until reaching goal
- Undiscounted

# Sarsa on the Windy Gridworld



# Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$
- While following behaviour policy $\mu(a|s)$

$$\{S_1, A_1, R_2, ..., S_T\} \sim \mu$$

- Why is this important?
- Learn from observing humans or other agents
- Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi_{t-1}$
- Learn about *optimal* policy while following *exploratory* policy
- Learn about *multiple* policies while following *one* policy

# Q-Learning

- We now consider off-policy learning of action-values $Q(s, a)$
- No importance sampling is required
- Next action is chosen using behaviour policy $A_{t+1} \sim \mu(\cdot|S_t)$
- But we consider alternative successor action $A' \sim \pi(\cdot|S_t)$
- And update $Q(S_t, A_t)$ towards value of alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left( R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t) \right)$$

# Off-Policy Control with Q-Learning

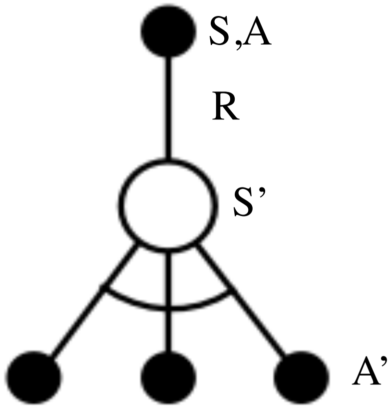
- We now allow both behaviour and target policies to improve
- The target policy $\pi$ is greedy w.r.t. $Q(s, a)$

$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a')$$

- The behaviour policy $\mu$ is e.g. $\epsilon$-greedy w.r.t. $Q(s, a)$
- The Q-learning target then simplifies:

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ =& R_{t+1} + \gamma Q(S_{t+1}, \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a')) \\ =& R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') \end{aligned}$$

# Q-Learning Control Algorithm




$$Q(S, A) \leftarrow Q(S, A) + \alpha \left( R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right)$$

**Theorem**

*Q-learning control converges to the optimal action-value function, $Q(s, a) \rightarrow q_*(s, a)$*

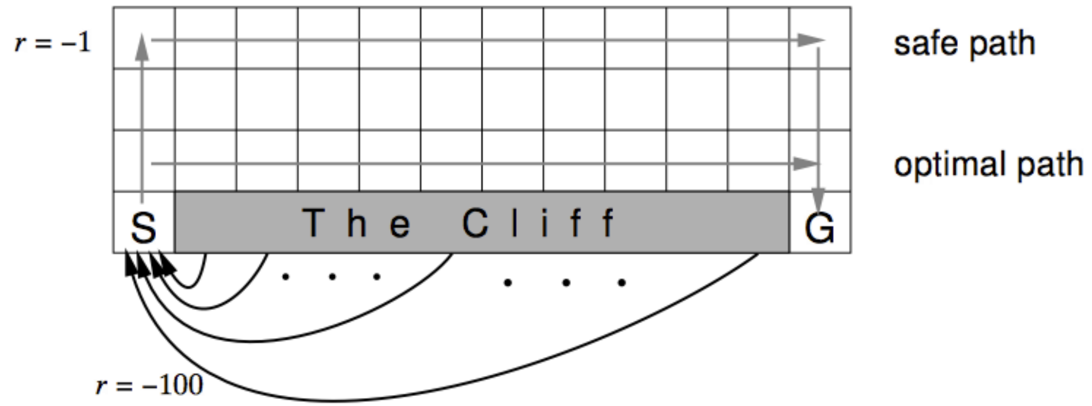
# Q-Learning Algorithm for Off-Policy Control

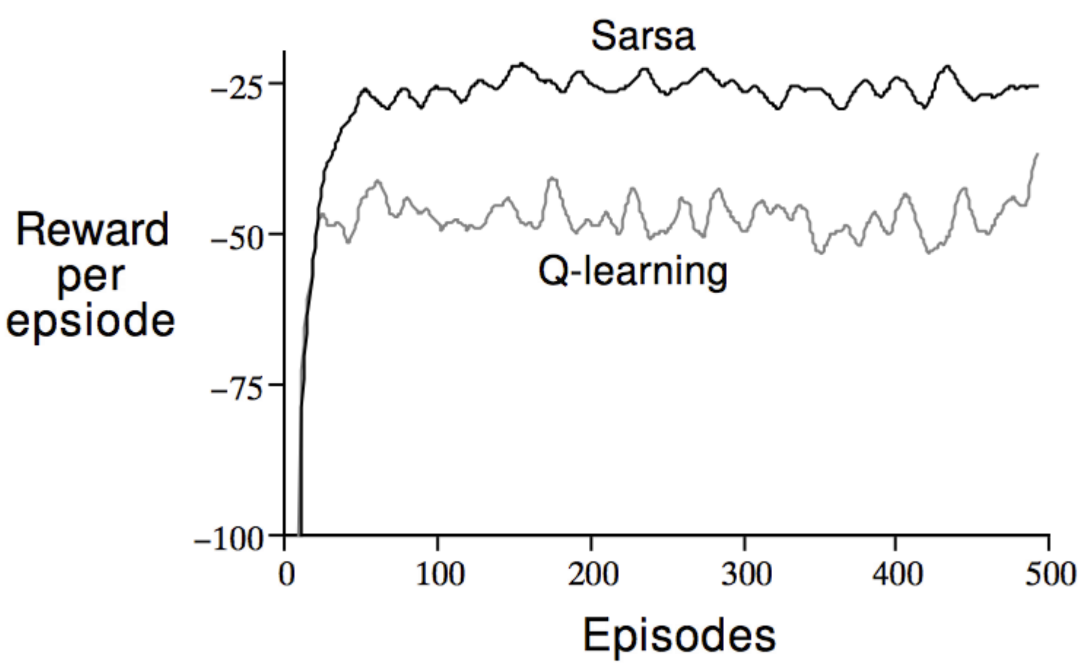
Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize $S$
    Repeat (for each step of episode):
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S, A) \leftarrow Q(S, A) + \alpha \left[ R + \gamma \max_a Q(S', a) - Q(S, A) \right]$
        $S \leftarrow S'$;
    until $S$ is terminal

# Q-Learning Demo

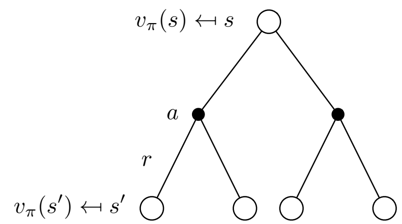
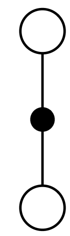
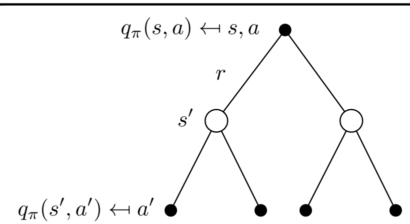
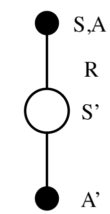
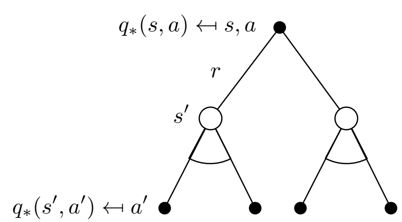
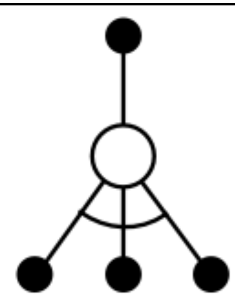
Q-Learning Demo

# Cliff Walking Example



# Relationship Between DP and TD

| | *Full Backup (DP)* | *Sample Backup (TD)* |
|---|---|---|
| Bellman Expectation Equation for $v_\pi(s)$ |   Iterative Policy Evaluation |  TD Learning |
| Bellman Expectation Equation for $q_\pi(s,a)$ |   Q-Policy Iteration |   Sarsa |
| Bellman Optimality Equation for $q_*(s,a)$ |   Q-Value Iteration |  Q-Learning |

# Relationship Between DP and TD (2)

| *Full Backup (DP)* | *Sample Backup (TD)* |
|---|---|
| Iterative Policy Evaluation | TD Learning |
| $V(s) \leftarrow \mathbb{E}\left[R + \gamma V(S') \mid s\right]$ | $V(S) \xleftarrow{\alpha} R + \gamma V(S')$ |
| Q-Policy Iteration | Sarsa |
| $Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(S',A') \mid s,a\right]$ | $Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$ |
| Q-Value Iteration | Q-Learning |
| $Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$ | $Q(S,A) \xleftarrow{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$ |

where $x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$