Decision Making and Reinforcement Learning

Module 6: Monte Carlo Methods

Tony Dear, Ph.D.

Department of Computer Science

School of Engineering and Applied Sciences

Topics

- Value estimation using sample averaging
- First-visit Monte Carlo prediction

- State-action values
- ε–greedy on-policy MC control

- Importance sampling
- Off-policy MC control

Learning Objectives

- **Estimate** state values using MC prediction
- **Estimate** state-action values using ε−greedy MC control and use them to learn or improve policies
- **Use** importance sampling to estimate values of different policies
- **Implement** off-policy MC control to learn optimal policies

Learning from Experience

- Dynamic programming requires knowledge of environment *model*
- Often inaccessible or difficult to compute (e.g., belief MDPs)

Learning from Experience

- Dynamic programming requires knowledge of environment *model*
- Often inaccessible or difficult to compute (e.g., belief MDPs)

- **Reinforcement learning**: Find optimal policies through experience
- Interact with environment, receive rewards, and formulate policies

Learning from Experience

- Dynamic programming requires knowledge of environment *model*
- Often inaccessible or difficult to compute (e.g., belief MDPs)

- **Reinforcement learning**: Find optimal policies through experience
- Interact with environment, receive rewards, and formulate policies

- Examples from real life
- How do humans learn? Other biological organisms?

Monte Carlo Methods

- **Monte Carlo** methods: Generate sampled experience and average them for different states and actions

Monte Carlo Methods

- **Monte Carlo** methods: Generate sampled experience and average them for different states and actions

- Recall the definition of value function for a fixed policy π:

$$V^{\pi}(s) = E\left[\sum_{t=0} \gamma^{t} R(s_t, \pi(s_t), s_{t+1})\right]$$

Monte Carlo Methods

- **Monte Carlo** methods: Generate sampled experience and average them for different states and actions

- Recall the definition of value function for a fixed policy π:

$$V^{\pi}(s) = E\left[\sum_{t=0} \gamma^{t} R(s_t, \pi(s_t), s_{t+1})\right]$$

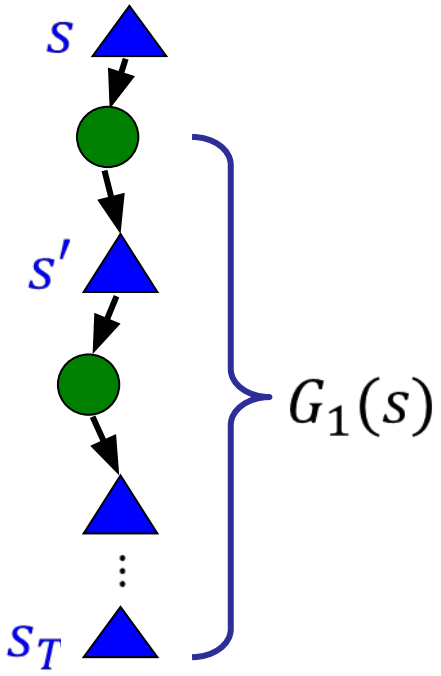
- Idea: Approximate the expectation by taking *averages* of sample reward sequences over multiple *episodes*

MC: State Value Estimation

- Idea: $V(s)$ can be estimated by averaging utilities observed *after* visiting s
- Think of each sample as a path from a given node to a leaf node in search tree

MC: State Value Estimation

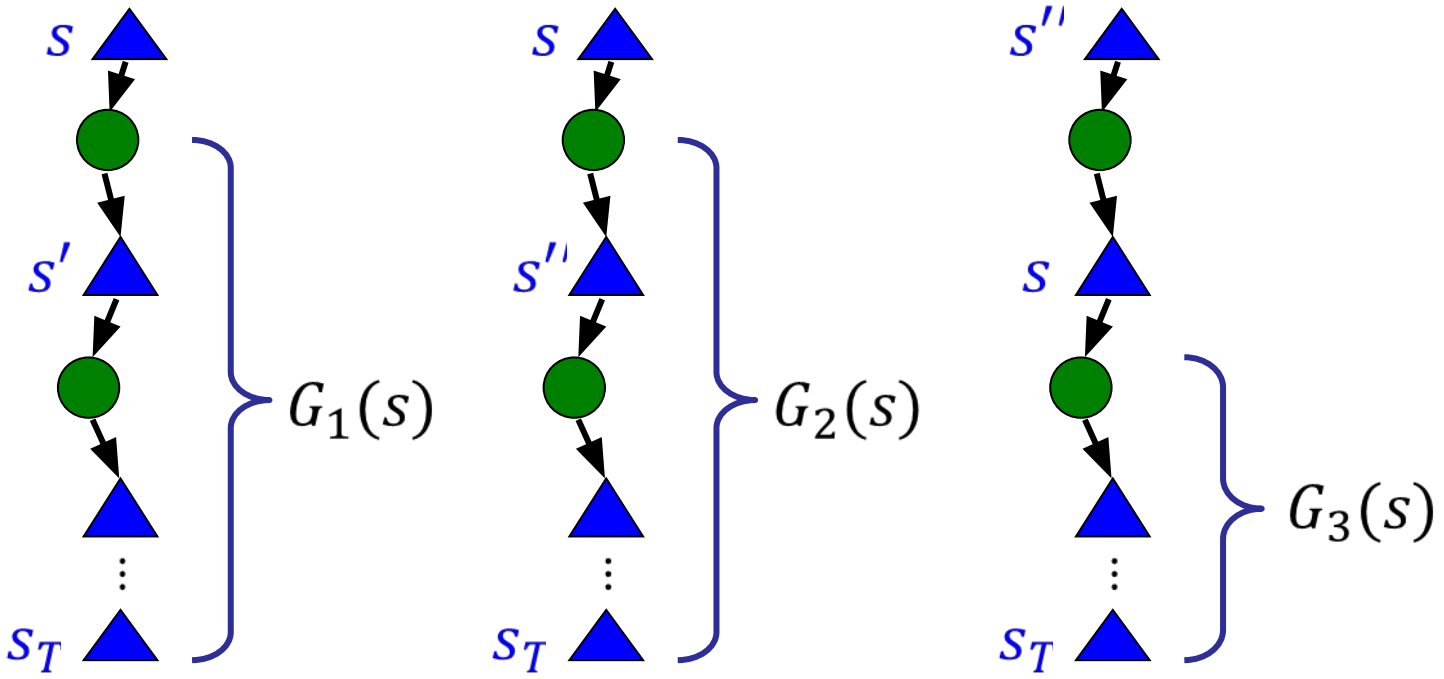
- Idea: $V(s)$ can be estimated by averaging utilities observed *after* visiting s
- Think of each sample as a path from a given node to a leaf node in search tree



Episode 1

MC: State Value Estimation

- Idea: $V(s)$ can be estimated by averaging utilities observed *after* visiting s
- Think of each sample as a path from a given node to a leaf node in search tree



Episode 1 Episode 2 Episode 3

$$\cdots \qquad V(s) = \frac{1}{N}\sum_{i=1}^{N} G_i(s)$$

First-Visit MC Prediction

- **Monte Carlo**: Estimate state values by averaging utilities over multiple episodes
- *First-visit* MC: A value is estimated after first visit to state within episode

First-Visit MC Prediction

- **Monte Carlo**: Estimate state values by averaging utilities over multiple episodes
- *First-visit* MC: A value is estimated after first visit to state within episode

- **Initialize** $V^{\pi}(s) \leftarrow 0, Count(s) \leftarrow 0$ for each state $s \in S$

First-Visit MC Prediction

- **Monte Carlo**: Estimate state values by averaging utilities over multiple episodes
- *First-visit* MC: A value is estimated after first visit to state within episode

- **Initialize** $V^{\pi}(s) \leftarrow 0, Count(s) \leftarrow 0$ for each state $s \in S$
- **Loop**:
  - **Generate** episode E following π: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$

First-Visit MC Prediction

- **Monte Carlo**: Estimate state values by averaging utilities over multiple episodes
- *First-visit* MC: A value is estimated after first visit to state within episode

- **Initialize** $V^{\pi}(s) \leftarrow 0, Count(s) \leftarrow 0$ for each state $s \in S$
- **Loop**:
  - **Generate** episode E following π: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each state $s_t \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}$

First-Visit MC Prediction

- **Monte Carlo**: Estimate state values by averaging utilities over multiple episodes
- *First-visit* MC: A value is estimated after first visit to state within episode

- **Initialize** $V^{\pi}(s) \leftarrow 0, Count(s) \leftarrow 0$ for each state $s \in S$
- **Loop**:
  - **Generate** episode E following π: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each state $s_t \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}$
    - $V^{\pi}(s_t) \leftarrow (Count(s_t) \times V^{\pi}(s_t) + G)/(Count(s_t) + 1)$

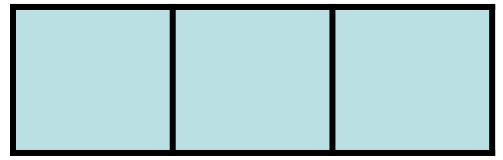
First-Visit MC Prediction

- **Monte Carlo**: Estimate state values by averaging utilities over multiple episodes
- *First-visit* MC: A value is estimated after first visit to state within episode

- **Initialize** $V^{\pi}(s) \leftarrow 0, Count(s) \leftarrow 0$ for each state $s \in S$
- **Loop**:
  - **Generate** episode E following π: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each state $s_t \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}$
    - $V^{\pi}(s_t) \leftarrow (Count(s_t) \times V^{\pi}(s_t) + G)/(Count(s_t) + 1)$
    - $Count(s_t) \leftarrow Count(s_t) + 1$

Example: Mini-Gridworld

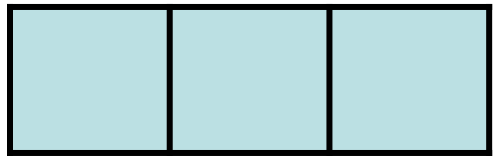
- States: A, B, C; actions: L, R; rewards received upon entering each state



$\gamma = 0.5$

Example: Mini-Gridworld

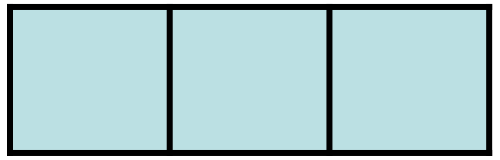
- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)



$\gamma = 0.5$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)

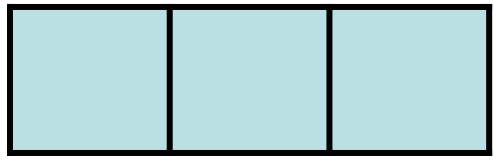


- Episode 1: $(A, +3, A, -2, B, +1, C, -2, B, +3)$

$\gamma = 0.5$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)



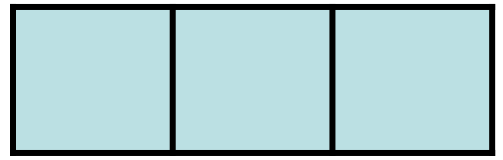
- Episode 1: $(A, +3, A, -2, B, +1, C, -2, B, +3)$

$\gamma = 0.5$

$$V^{\pi}(A) = 3 + \gamma(-2) + \gamma^2(1) + \gamma^3(-2) + \gamma^4(3) = 2.1875$$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)



- Episode 1: $(A, +3, A, -2, B, +1, C, -2, B, +3)$

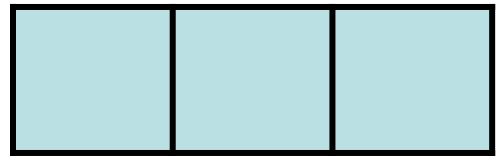
$\gamma = 0.5$

$$V^{\pi}(A) = 3 + \gamma(-2) + \gamma^2(1) + \gamma^3(-2) + \gamma^4(3) = 2.1875$$

$$V^{\pi}(B) = 1 + \gamma(-2) + \gamma^2(3) = 0.75$$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)



- Episode 1: $(A, +3, A, -2, B, +1, C, -2, B, +3)$

$\gamma = 0.5$

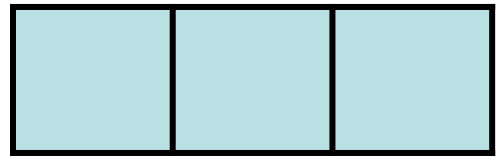
$$V^{\pi}(A) = 3 + \gamma(-2) + \gamma^2(1) + \gamma^3(-2) + \gamma^4(3) = 2.1875$$

$$V^{\pi}(B) = 1 + \gamma(-2) + \gamma^2(3) = 0.75$$

$$V^{\pi}(C) = -2 + \gamma(3) = -0.5$$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)



- Episode 2: $(A, -2, B, +3, A, -2, B, +1, C, -2)$

$\gamma = 0.5$

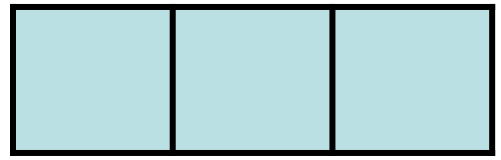
$$G(A) = -1$$

$$G(B) = 2$$

$$G(C) = -2$$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)



- Episode 2: $(A, -2, B, +3, A, -2, B, +1, C, -2)$ $\gamma = 0.5$

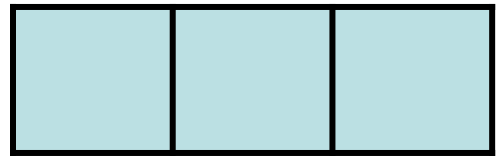
$$G(A) = -1 \qquad V^{\pi}(A) \leftarrow \frac{1}{2}\left(V^{\pi}(A) + G(A)\right) = 0.59375$$

$$G(B) = 2 \qquad V^{\pi}(B) \leftarrow \frac{1}{2}\left(V^{\pi}(B) + G(B)\right) = 1.375$$

$$G(C) = -2 \qquad V^{\pi}(C) \leftarrow \frac{1}{2}\left(V^{\pi}(C) + G(C)\right) = -1.25$$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R; rewards received upon entering each state
- Policy: $\pi(s) = L$ for all states s
- Each episode ends after 5 actions (finite-horizon)



- Episode 3: $(C, +1, C, -2, B, +3, A, -2, B, +3)$

$\gamma = 0.5$

$$G(A) = -0.5 \qquad V^{\pi}(A) \leftarrow \frac{1}{3}\left(2V^{\pi}(A) + G(A)\right) = 0.229$$

$$G(B) = 2.75 \qquad V^{\pi}(B) \leftarrow \frac{1}{3}\left(2V^{\pi}(B) + G(B)\right) = 1.833$$

$$G(C) = 0.6875 \qquad V^{\pi}(C) \leftarrow \frac{1}{3}\left(2V^{\pi}(C) + G(C)\right) = -0.604$$

Finer Points

- Some states may be visited more often than others
- Values converge to true V^{π} after many, many visits

Finer Points

- Some states may be visited more often than others
- Values converge to true V^{π} after many, many visits

- Estimates of different state values are *independent* (in contrast to DP)
- Accuracy of $V^{\pi}(s)$ does *not* depend on accuracy of $V^{\pi}(s')$

Finer Points

- Some states may be visited more often than others
- Values converge to true V^{π} after many, many visits

- Estimates of different state values are *independent* (in contrast to DP)
- Accuracy of $V^{\pi}(s)$ does *not* depend on accuracy of $V^{\pi}(s')$

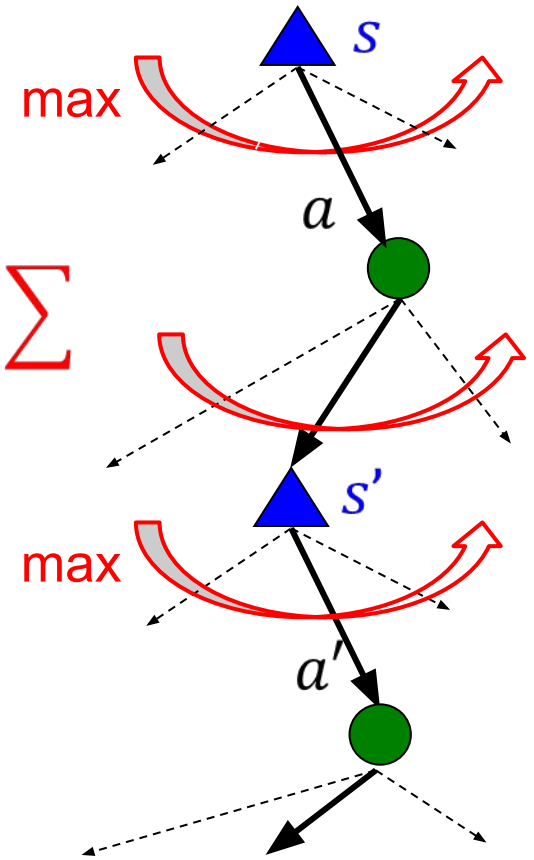
- Result: Computational complexity of estimating specific state values is independent of state space size!
- Can choose to focus on certain states and ignore others

State-Action Values

- Suppose we are given a value function V^{π}
- How do we compute the associated policy π?

$$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')]$$

don't have these!



State-Action Values

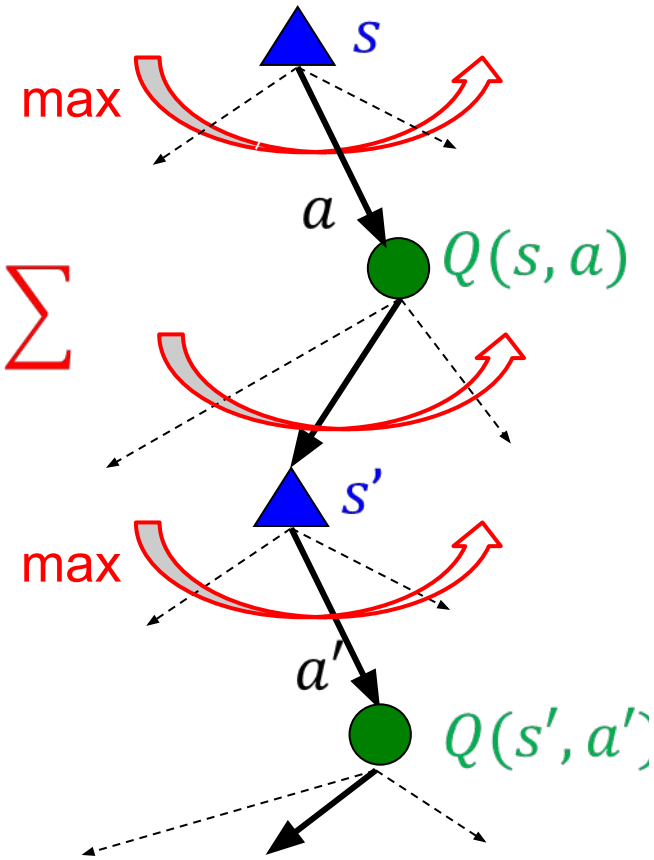
- Suppose we are given a value function V^{π}
- How do we compute the associated policy π?

$$\pi(s) \leftarrow \operatorname*{argmax}_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')]$$

don't have these!

- Idea: *Learn* **state-action values** values at the chance nodes

$$\pi(s) \leftarrow \operatorname*{argmax}_a Q(s,a)$$



State-Action Values

- Suppose we are given a value function V^{π}
- How do we compute the associated policy π?

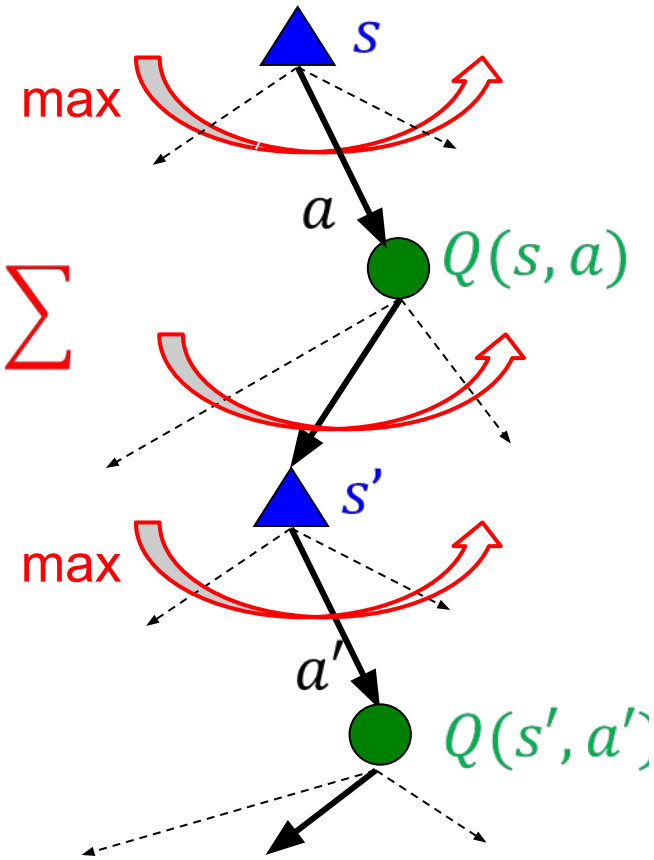
$$\pi(s) \leftarrow \operatorname*{argmax}_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')]$$

don't have these!

- Idea: *Learn* **state-action values** values at the chance nodes

$$\pi(s) \leftarrow \operatorname*{argmax}_a Q(s,a)$$

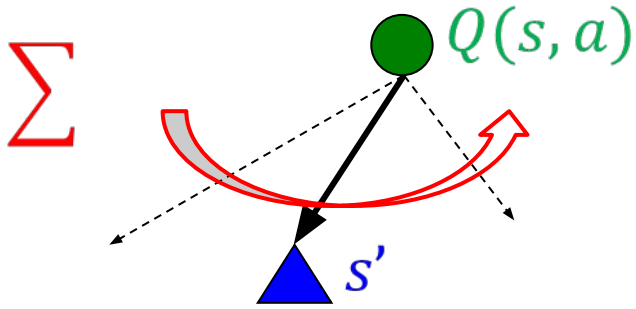
- $Q(s,a)$ is the *expected* utility of s after taking action a and then following π thereafter



Q Values, State Values, and Policy

- **Q values** can be defined as functions of V^* or recursively as functions of themselves

$$Q(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

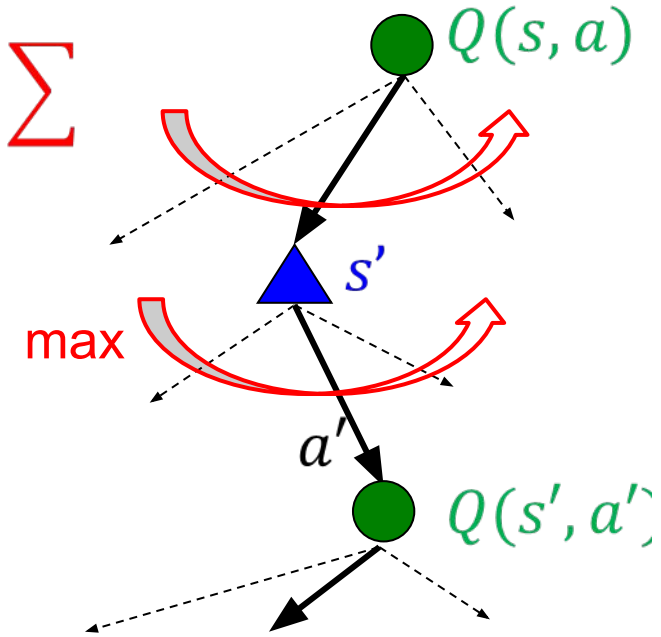


Q Values, State Values, and Policy

- **Q values** can be defined as functions of V^* or recursively as functions of themselves

$$Q(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

$$= \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q(s',a')\right]$$

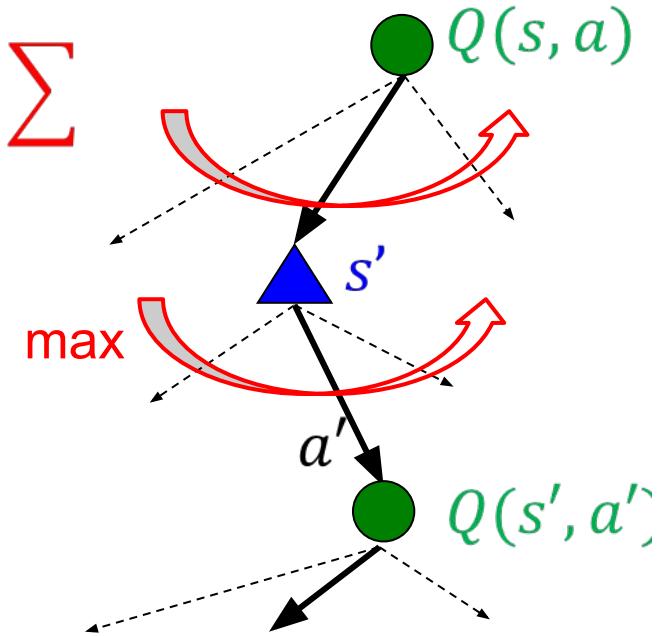


Q Values, State Values, and Policy

- **Q values** can be defined as functions of V^* or recursively as functions of themselves

$$Q(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

$$= \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q(s',a')\right]$$



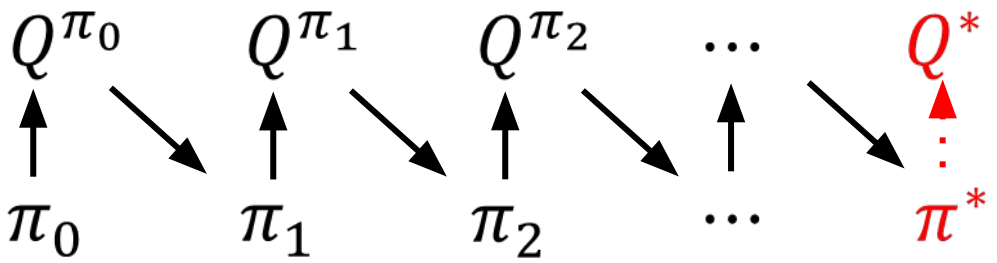
- To learn policies using RL, we will first learn Q values
- Then use these to compute V^* or π^*

$$V^*(s) = \max_a Q(s,a)$$

$$\pi^*(s) = \operatorname{argmax}_a Q(s,a)$$

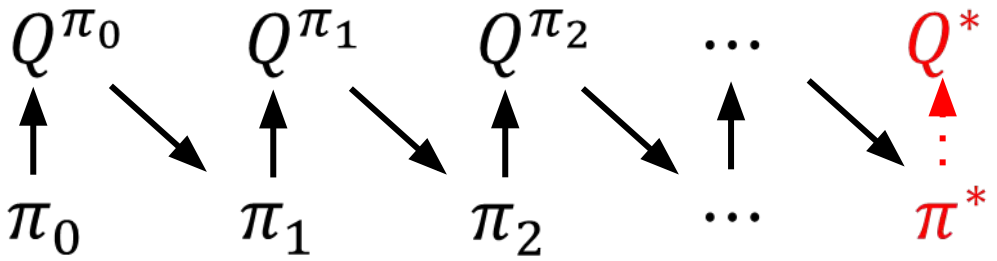
Monte Carlo Control

- Goal: Learn optimal (or close to optimal) policy via Monte Carlo RL
- Idea from policy iteration: Alternate between evaluation and improvement



Monte Carlo Control

- Goal: Learn optimal (or close to optimal) policy via Monte Carlo RL
- Idea from policy iteration: Alternate between evaluation and improvement



- Evaluation can be done using MC prediction with one or more episodes
- Improvement: Select new actions greedily

$$\pi_{i+1}(s) = \underset{a}{\operatorname{argmax}}\, Q^{\pi_i}(s, a)$$

Convergence and Exploration

- Problem: Policy evaluation requires many episodes
- Instead of waiting for current policy values to converge, we can do one evaluation and one improvement step per episode

- As in generalized policy iteration, this will converge to optimal values and policy
- Argmax operation can only improve current policy

- When learning a new policy we need to experience all relevant state-action pairs
- **Exploring starts**: Choose a random state-action to start in each episode
- Theoretical convergence after infinite episodes

ε-Greedy Policies

- How to ensure that we correctly learn Q values for all state-action pairs?
- We can choose different or random starting states in each episode

ε-Greedy Policies

- How to ensure that we correctly learn Q values for all state-action pairs?
- We can choose different or random starting states in each episode

- Another idea: Incorporate *exploration* into MC samples
- **ε-greedy policy**: Choose best action most of the time, but with small probability ε, execute random action instead

ε-Greedy Policies

- How to ensure that we correctly learn Q values for all state-action pairs?
- We can choose different or random starting states in each episode

- Another idea: Incorporate *exploration* into MC samples
- **ε-greedy policy**: Choose best action most of the time, but with small probability ε, execute different, random action instead

Stochastic policy:

$$\pi(a|s) = \begin{cases} 1 - \varepsilon & \text{for } a = \underset{a'}{\operatorname{argmax}}\, Q(s, a') \\ \dfrac{\varepsilon}{|A(s)| - 1} & \text{for all other actions } a \end{cases}$$

ε–Greedy MC Control

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, Count(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $\varepsilon \leftarrow$ small number, $\pi \leftarrow$ an ε-greedy policy

ε–Greedy MC Control

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, Count(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $\varepsilon \leftarrow$ small number, $\pi \leftarrow$ an ε-greedy policy

- **Loop**:
  - **Generate** new episode E following π: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each pair $(s_t, a_t) \in E$:

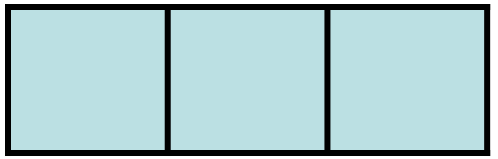
ε–Greedy MC Control

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, Count(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $\varepsilon \leftarrow$ small number, $\pi \leftarrow$ an ε-greedy policy

- **Loop**:
  - **Generate** new episode E following π: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each pair $(s_t, a_t) \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}$
    - $Q^{\pi}(s_t, a_t) \leftarrow (Count(s_t, a_t) \times Q^{\pi}(s_t, a_t) + G)/(Count(s_t, a_t) + 1)$
    - $Count(s_t, a_t) \leftarrow Count(s_t, a_t) + 1$

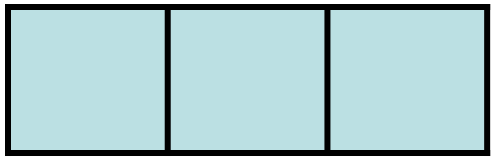
Example: Mini-Gridworld

- States: A, B, C; actions: L, R
- All Q values initialized to 0; $\gamma = 0.8$



Example: Mini-Gridworld

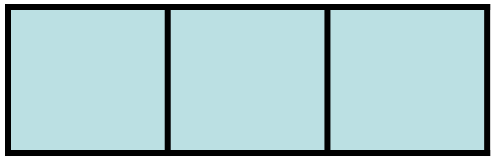
- States: A, B, C; actions: L, R
- All Q values initialized to 0; $\gamma = 0.8$



- Episode: $A, L, +3, A, L, -2, B, R, +1, C, L, -2$
- MC prediction: $Q(A, R), Q(B, L), Q(C, R)$ are unchanged (stay at 0)

Example: Mini-Gridworld

- States: A, B, C; actions: L, R
- All Q values initialized to 0; $\gamma = 0.8$



- Episode: $A, L, +3, A, L, -2, B, R, +1, C, L, -2$
- MC prediction: $Q(A, R), Q(B, L), Q(C, R)$ are unchanged (stay at 0)
- Following Q values change:

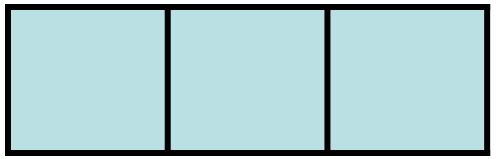
$$Q(A, L) = 3 + \gamma(-2) + \gamma^2(1) + \gamma^3(-2) = 1.016$$

$$Q(B, R) = 1 + \gamma(-2) = -0.6$$

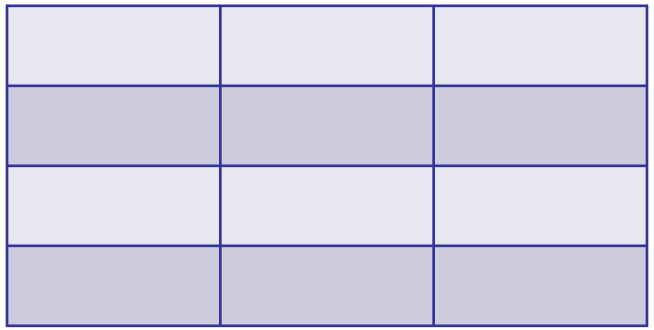
$$Q(C, L) = -2$$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R
- All Q values initialized to 0; $\gamma = 0.8$

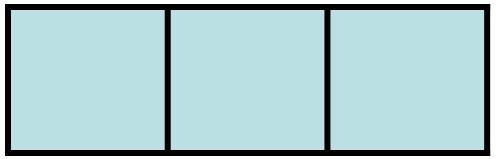


- Episode: $A, L, +3, A, L, -2, B, R, +1, C, L, -2$

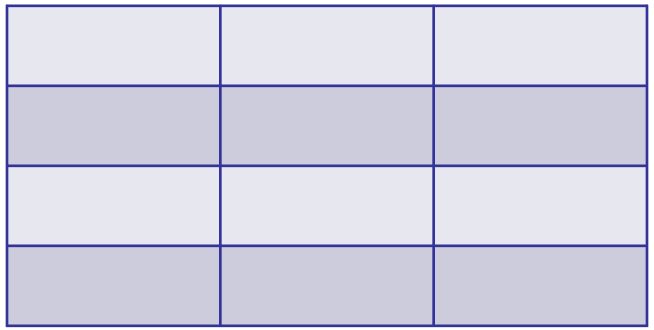


Example: Mini-Gridworld

- States: A, B, C; actions: L, R
- All Q values initialized to 0; $\gamma = 0.8$



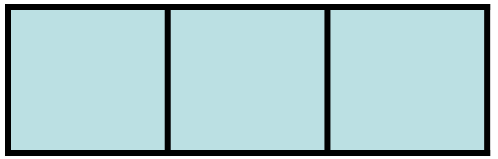
- Episode: $A, L, +3, A, L, -2, B, R, +1, C, L, -2$



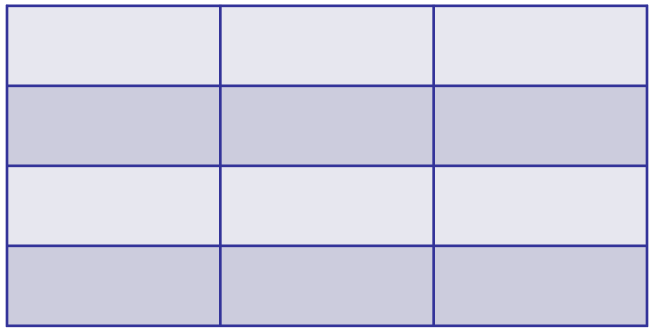
- New policy and probabilities:
- Greedy action probabilities: $\pi(L|A) = \pi(L|B) = \pi(R|C) = 1 - \varepsilon$

Example: Mini-Gridworld

- States: A, B, C; actions: L, R
- All Q values initialized to 0; $\gamma = 0.8$



- Episode: $A, L, +3, A, L, -2, B, R, +1, C, L, -2$



- New policy and probabilities:

- Greedy action probabilities: $\pi(L|A) = \pi(L|B) = \pi(R|C) = 1 - \varepsilon$
- Opposite action probabilities: $\pi(R|A) = \pi(R|B) = \pi(L|C) = \varepsilon$

On-Policy Learning

- Problem: ε–greedy MC control doesn't actually give us Q and π^*

- Since we have $\varepsilon > 0$ and occasionally explore while learning, the state-action values and policy converge *under that condition*

On-Policy Learning

- Problem: ε–greedy MC control doesn't actually give us Q and π^*
- Since we have $\varepsilon > 0$ and occasionally explore while learning, the state-action values and policy converge *under that condition*
- **On-policy** learning: Learn the values for the policy that we are following, or the *behavior policy*
- If we wanted the optimal policy, we would have to decrease ε over time

Off-Policy Learning

- **Off-policy** learning: Use the *behavior policy* b to learn about a different *target policy* π

- More general than on-policy learning, but slower to converge

Off-Policy Learning

- **Off-policy** learning: Use the *behavior policy* b to learn about a different *target policy* π
- More general than on-policy learning, but slower to converge
- Off-policy *prediction*: Estimate values for π using episodes from b
- We require that all state-actions allowed by π also allowed by b: $\pi(a_j|s_j) > 0 \Rightarrow b(a_j|s_j) > 0$

Importance Sampling

- Suppose we observe an episode from behavior policy b:
  $s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, \dots, s_{T-1}, a_{T-1}, r_T$

Importance Sampling

- Suppose we observe an episode from behavior policy b:
  $S_t, A_t, S_{t+1}, A_{t+1}, \dots, S_{T-1}, A_{T-1}$

- **Importance sampling ratio**: *Relative likelihood* of subset of states and actions under different policy π

$$w_t = \prod_{j=t}^{T-1} \frac{\pi(a_j|s_j)}{b(a_j|s_j)}$$

Importance Sampling

- Suppose we observe an episode from behavior policy b:
  $s_t, a_t, s_{t+1}, a_{t+1}, \dots, s_{T-1}, a_{T-1}$

- **Importance sampling ratio**: *Relative likelihood* of subset of states and actions under different policy π

$$w_t = \prod_{j=t}^{T-1} \frac{\pi(a_j|s_j)}{b(a_j|s_j)}$$

- We can this weight to transform the value of a state-action pair under b into one as if generated by π

Weighted Averages

- Recall that computation of G_t, the state-action utility of (s_t, a_t), comes from the rewards seen *after* seeing (s_t, a_t)

Weighted Averages

- Recall that computation of G_t, the state-action utility of (s_t, a_t), comes from the rewards seen *after* seeing (s_t, a_t)

- To transform G_t into its equivalent value in a different policy, *multiply* it by the importance sampling ratio w_{t+1}

$$Q(s_t, a_t) = E[w_{t+1} G_t]$$

Weighted Averages

- Recall that computation of G_t, the state-action utility of (s_t, a_t), comes from the rewards seen *after* seeing (s_t, a_t)

- To transform G_t into its equivalent value in a different policy, *multiply* it by the importance sampling ratio w_{t+1}

$$Q(s_t, a_t) = E[w_{t+1} G_t]$$

- When updating $Q(s_t, a_t)$ with estimates from multiple episodes, we compute a *weighted average* by dividing by the sum of all weights

Off-Policy MC Control with Weighted IS

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, W(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $b \leftarrow \varepsilon$-greedy behavior policy, $\pi \leftarrow$ greedy policy

Off-Policy MC Control with Weighted IS

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, W(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $b \leftarrow \varepsilon$-greedy behavior policy, $\pi \leftarrow$ greedy policy

- **Loop**:
  - **Generate** new episode E following b: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each pair $(s_t, a_t) \in E$:

Off-Policy MC Control with Weighted IS

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, W(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $b \leftarrow \varepsilon$-greedy behavior policy, $\pi \leftarrow$ greedy policy

- **Loop**:
  - **Generate** new episode E following b: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each pair $(s_t, a_t) \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}, w \leftarrow \prod_{j=t+1}^{T-1} \frac{\pi(a_j|s_j)}{b(a_j|s_j)}$

Off-Policy MC Control with Weighted IS

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, W(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $b \leftarrow \varepsilon$-greedy behavior policy, $\pi \leftarrow$ greedy policy

- **Loop**:
  - **Generate** new episode E following b: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each pair $(s_t, a_t) \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}$, $w \leftarrow \prod_{j=t+1}^{T-1} \frac{\pi(a_j|s_j)}{b(a_j|s_j)}$
    - $Q^{\pi}(s_t, a_t) \leftarrow (W(s_t, a_t) \times Q^{\pi}(s_t, a_t) + w \times G)/(W(s_t, a_t) + w)$

Off-Policy MC Control with Weighted IS

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, W(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $b \leftarrow \varepsilon$-greedy behavior policy, $\pi \leftarrow$ greedy policy

- **Loop**:
  - **Generate** new episode E following b: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each pair $(s_t, a_t) \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}$, $w \leftarrow \prod_{j=t+1}^{T-1} \frac{\pi(a_j|s_j)}{b(a_j|s_j)}$
    - $Q^{\pi}(s_t, a_t) \leftarrow (W(s_t, a_t) \times Q^{\pi}(s_t, a_t) + w \times G)/(W(s_t, a_t) + w)$
    - $W(s_t, a_t) \leftarrow W(s_t, a_t) + w$

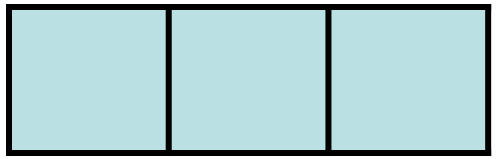
Off-Policy MC Control with Weighted IS

- **Initialize** $Q^{\pi}(s,a) \leftarrow 0, W(s,a) \leftarrow 0, \forall s \in S, \forall a \in A$
- **Initialize** $b \leftarrow \varepsilon$-greedy behavior policy, $\pi \leftarrow$ greedy policy

- **Loop**:
  - **Generate** new episode E following b: $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** first occurrence of each pair $(s_t, a_t) \in E$:
    - $G \leftarrow \sum_{j=0}^{T-t-1} \gamma^j r_{j+t+1}$, $w \leftarrow \prod_{j=t+1}^{T-1} \frac{\pi(a_j|s_j)}{b(a_j|s_j)}$
    - $Q^{\pi}(s_t, a_t) \leftarrow (W(s_t, a_t) \times Q^{\pi}(s_t, a_t) + w \times G)/(W(s_t, a_t) + w)$
    - $W(s_t, a_t) \leftarrow W(s_t, a_t) + w$
    - $\pi(s_t) \leftarrow \operatorname{argmax} Q^{\pi}(s,a)$

Example: Mini-Gridworld

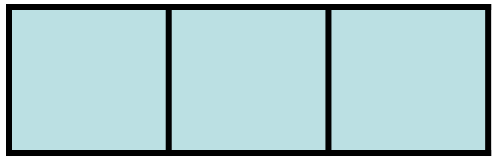
- All Q values initialized to 0; $\gamma = 0.8$



- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



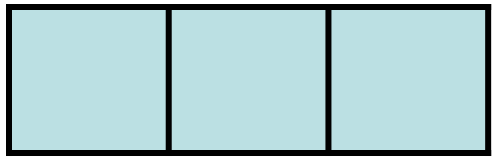
- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 1: $B, R, +1, C, L, -2, B, L, +1$

	Episode 1	Episode 2	

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



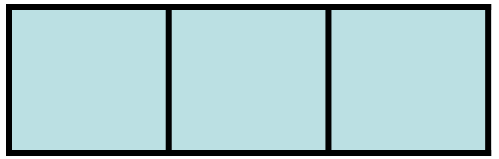
- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 1: $B, R, +1, C, L, -2, B, L, +1$

	Episode 1	Episode 2	

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



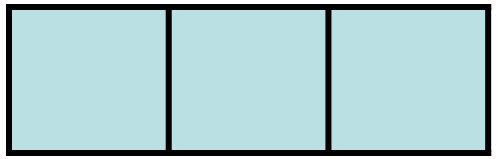
- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 1: $B, R, +1, C, L, -2, B, L, +1$

	Episode 1	Episode 2	

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



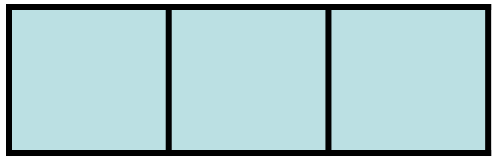
- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 2: $C, L, -2, B, R, +3, A, L, +3$

	Episode 1	Episode 2	
	Not updated		

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



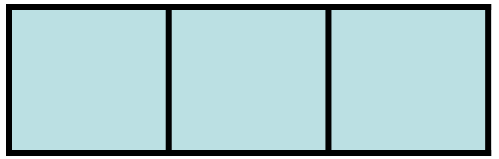
- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 2: $C, L, -2, B, R, +3, A, L, +3$

	Episode 1	Episode 2	
	Not updated		

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



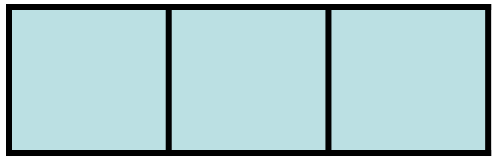
- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 2: $C, L, -2, B, R, +3, A, L, +3$

	Episode 1	Episode 2	
	Not updated		

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



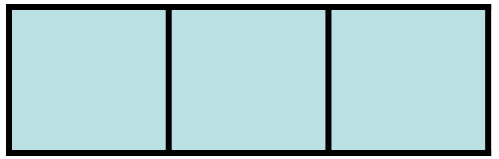
- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 2: $C, L, -2, B, R, +3, A, L, +3$

	Episode 1	Episode 2	
	Not updated		

Example: Mini-Gridworld

- All Q values initialized to 0; $\gamma = 0.8$



- Target policy: $\pi(s) = \text{argmax}_{a'} Q(s, a')$; break ties with L
- Behavior policy: $b(\pi(s)|s) = 0.8$ for all states

- Episode 2: $C, L, -2, B, R, +3, A, L, +3$

	Episode 1	Episode 2	
	Not updated		
		Not updated	

Convergence Properties

- If π is deterministic (or has any $\pi(a|s) = 0$), the IS ratio of any sequence containing **exploratory** (off-policy) actions will be 0

Convergence Properties

- If π is deterministic (or has any $\pi(a|s) = 0$), the IS ratio of any sequence containing **exploratory** (off-policy) actions will be 0

- Can make MC control more efficient by discarding **first part** of episode **prior to last exploratory action**

Convergence Properties

- If π is deterministic (or has any $\pi(a|s) = 0$), the IS ratio of any sequence containing **exploratory** (off-policy) actions will be 0

- Can make MC control more efficient by discarding **first part** of episode **prior to last exploratory action**

- Convergence to optimal values will occur in the limit
- Can be slow if exploration is too common
- Experience from beginning of episodes is often wasted

Summary

- MC methods learn value functions and policies from experience
- Useful when model is unknown or difficult to compute
- Useful for focusing on subset of state space

- MC prediction averages utilities from episode sequences to evaluate policies
- Exploration is required for MC control and learning new policies

- *On-policy* methods such as ε-greedy optimize policies that maintain exploration
- *Off-policy* methods use importance sampling to transform utilities generated by a *behavior policy* to estimate those of a *target policy*