

# Lecture 4: Model Free Control

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2021

- Structure closely follows much of David Silver's Lecture 5. For additional reading please see SB Sections 5.2-5.4, 6.4, 6.5, 6.7

# Refresh Your Knowledge 3. Piazza Poll

- Which of the following equations express a TD update?
  - ①  $V(s_t) = r(s_t, a_t) + \gamma \sum_{s'} p(s'|s_t, a_t) V(s')$
  - ②  $V(s_t) = (1 - \alpha)V(s_t) + \alpha(r(s_t, a_t) + \gamma V(s_{t+1}))$
  - ③  $V(s_t) = (1 - \alpha)V(s_t) + \alpha \sum_{i=t}^H r(s_i, a_i)$
  - ④  $V(s_t) = (1 - \alpha)V(s_t) + \alpha \max_a (r(s_t, a) + \gamma V(s_{t+1}))$
  - ⑤ Not sure
- Bootstrapping is
  - ① When samples of  $(s, a, s')$  transitions are used to approximate the true expectation over next states
  - ② When an estimate of the next state value is used instead of the true next state value
  - ③ Used in Monte-Carlo policy evaluation
  - ④ Not sure

# Refresh Your Knowledge 3. Piazza Poll

- Which of the following equations express a TD update?  
True.  $V(s_t) = (1 - \alpha)V(s_t) + \alpha(r(s_t, a_t) + \gamma V(s_{t+1}))$
- Bootstrapping is when:  
An estimate of the next state value is used instead of the true next state value

# Break

# Class Structure

- Last time: Policy evaluation with no knowledge of how the world works (MDP model not given)
- This time: Control (making decisions) without a model of how the world works
- Next time: Generalization – Value function approximation

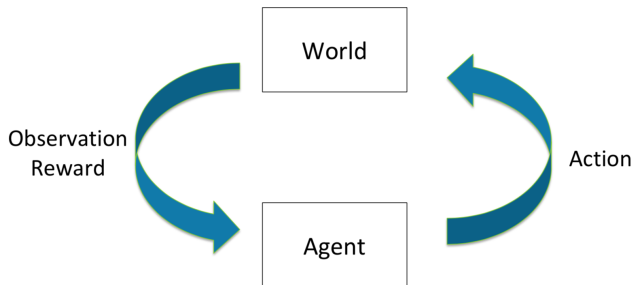
# Today: Model-free Control

- Generalized policy improvement
- Importance of exploration
- Monte Carlo control
- Model-free control with temporal difference (SARSA, Q-learning)
- Maximization bias

# Today: Model-free Control

- **Generalized policy improvement**
- **Importance of exploration**
- Monte Carlo control
- Model-free control with temporal difference (SARSA, Q-learning)
- Maximization bias

# Reinforcement Learning



- Goal: Learn to select actions to maximize total expected future reward



# Model-free Control Examples

- Many applications can be modeled as a MDP: Backgammon, Go, Robot locomotion, Helicopter flight, Robocup soccer, Autonomous driving, Customer ad selection, Invasive species management, Patient treatment
- For many of these and other problems either:
  - MDP model is unknown but can be sampled
  - MDP model is known but it is computationally infeasible to use directly, except through sampling

# Recall Policy Iteration

- Initialize policy  $\pi$
- Repeat:
  - Policy evaluation: compute  $V^\pi$
  - Policy improvement: update  $\pi$

$$\pi'(s) = \arg \max_a R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^\pi(s') = \arg \max_a Q^\pi(s, a)$$

- Now want to do the above two steps without access to the true dynamics and reward models
- Last lecture introduced methods for model-free policy evaluation

# Model Free Policy Iteration

- Initialize policy  $\pi$
- Repeat:
  - Policy evaluation: compute  $Q^\pi$
  - Policy improvement: update  $\pi$

# Model-free Generalized Policy Improvement

- Given an estimate  $Q^{\pi_i}(s, a) \forall s, a$
- Update new policy

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

# Check Your Understanding: Model-free Generalized Policy Improvement

- Given an estimate  $Q^{\pi_i}(s, a) \forall s, a$
- Update new policy

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

- Question: is this  $\pi_{i+1}$  deterministic or stochastic?
- Answer: Deterministic, Stochastic, Not Sure
- Recall in model-free policy evaluation, we estimated  $V^\pi$  by using  $\pi$  to generate new trajectories
- Question: Can we compute  $Q^{\pi_{i+1}}(s, a) \forall s, a$  by using this  $\pi_{i+1}$  to generate new trajectories?
- Answer: True, False, Not Sure

# Check Your Understanding: Model-free Generalized Policy Improvement

- Given an estimate  $Q^{\pi_i}(s, a) \forall s, a$
- Update new policy

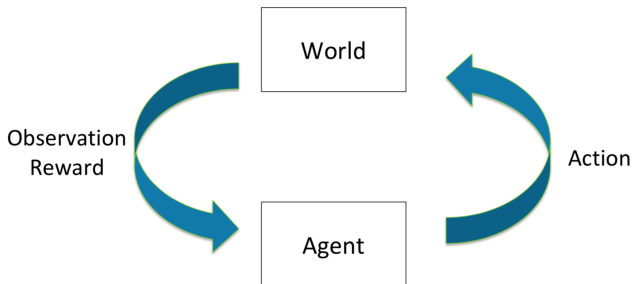
$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

- Question: is this  $\pi_{i+1}$  deterministic or stochastic?
- :Answer: Deterministic
- Recall in model-free policy evaluation, we estimated  $V^\pi$  by using  $\pi$  to generate new trajectories
- Question: Can we compute  $Q^{\pi_{i+1}}(s, a) \forall s, a$  by using this  $\pi_{i+1}$  to generate new trajectories?
- Answer: No.

# Model-free Policy Iteration

- Initialize policy  $\pi$
- Repeat:
  - Policy evaluation: compute  $Q^\pi$
  - Policy improvement: update  $\pi$  given  $Q^\pi$
- May need to modify policy evaluation:
  - If  $\pi$  is deterministic, can't compute  $Q(s, a)$  for any  $a \neq \pi(s)$
- How to interleave policy evaluation and improvement?
  - Policy improvement is now using an estimated  $Q$

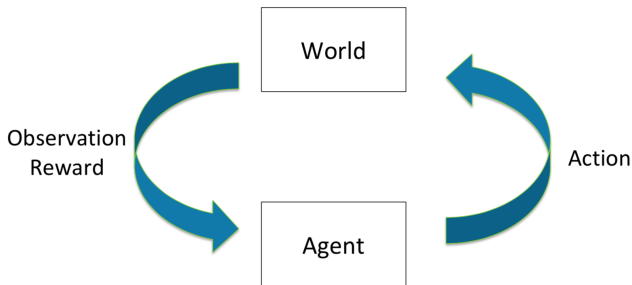
# The Problem of Exploration



- Goal: Learn to select actions to maximize total expected future reward
- Problem: Can't learn about actions without trying them
- Problem: But if we try new actions, spending less time taking actions that our past experience suggests will yield high reward



# Explore vs Exploit



- Explore: take actions haven't tried much in a state
- Exploit: take actions estimate will yield high discounted expected reward
- We will discuss this much more later in the course

# Policy Evaluation with Exploration

- Want to compute a model-free estimate of  $Q^\pi$
- In general seems subtle
  - Need to try all  $(s, a)$  pairs but then follow  $\pi$
  - Want to ensure resulting estimate  $Q^\pi$  is good enough so that policy improvement is a monotonic operator
- For certain classes of policies can ensure all  $(s, a)$  pairs are tried such that asymptotically  $Q^\pi$  converges to the true value

# $\epsilon$ -greedy Policies

- Simple idea to balance exploration and exploitation
- Let  $|A|$  be the number of actions
- Then an  $\epsilon$ -greedy policy w.r.t. a state-action value  $Q(s, a)$  is  $\pi(a|s) =$

# $\epsilon$ -greedy Policies

- Simple idea to balance exploration and exploitation
- Let  $|A|$  be the number of actions
- Then an  $\epsilon$ -greedy policy w.r.t. a state-action value  $Q(s, a)$  is  $\pi(a|s) =$ 
  - $\arg \max_a Q(s, a)$ , w. prob  $1 - \epsilon + \frac{\epsilon}{|A|}$
  - $a' \neq \arg \max Q(s, a)$  w. prob  $\frac{\epsilon}{|A|}$

# Monotonic $\epsilon$ -greedy Policy Improvement

## Theorem

For any  $\epsilon$ -greedy policy  $\pi_i$ , the  $\epsilon$ -greedy policy w.r.t.  $Q^{\pi_i}$ ,  $\pi_{i+1}$  is a monotonic improvement  $V^{\pi_{i+1}} \geq V^{\pi_i}$

$$\begin{aligned} Q^{\pi_i}(s, \pi_{i+1}(s)) &= \sum_{a \in A} \pi_{i+1}(a|s) Q^{\pi_i}(s, a) \\ &= (\epsilon/|A|) \left[ \sum_{a \in A} Q^{\pi_i}(s, a) \right] + (1 - \epsilon) \max_a Q^{\pi_i}(s, a) \end{aligned}$$

# Monotonic $\epsilon$ -greedy Policy Improvement

## Theorem

For any  $\epsilon$ -greedy policy  $\pi_i$ , the  $\epsilon$ -greedy policy w.r.t.  $Q^{\pi_i}$ ,  $\pi_{i+1}$  is a monotonic improvement  $V^{\pi_{i+1}} \geq V^{\pi_i}$

$$\begin{aligned} Q^{\pi_i}(s, \pi_{i+1}(s)) &= \sum_{a \in A} \pi_{i+1}(a|s) Q^{\pi_i}(s, a) \\ &= (\epsilon/|A|) \left[ \sum_{a \in A} Q^{\pi_i}(s, a) \right] + (1 - \epsilon) \max_a Q^{\pi_i}(s, a) \\ &= (\epsilon/|A|) \left[ \sum_{a \in A} Q^{\pi_i}(s, a) \right] + (1 - \epsilon) \max_a Q^{\pi_i}(s, a) \frac{1 - \epsilon}{1 - \epsilon} \\ &= (\epsilon/|A|) \left[ \sum_{a \in A} Q^{\pi_i}(s, a) \right] + (1 - \epsilon) \max_a Q^{\pi_i}(s, a) \sum_{a \in A} \frac{\pi_i(a|s) - \frac{\epsilon}{|A|}}{1 - \epsilon} \\ &\geq \frac{\epsilon}{|A|} \left[ \sum_{a \in A} Q^{\pi_i}(s, a) \right] + (1 - \epsilon) \sum_{a \in A} \frac{\pi_i(a|s) - \frac{\epsilon}{|A|}}{1 - \epsilon} Q^{\pi_i}(s, a) \\ &= \sum_{a \in A} \pi_i(a|s) Q^{\pi_i}(s, a) = V^{\pi_i}(s) \end{aligned}$$

- Therefore  $V^{\pi_{i+1}} \geq V^{\pi_i}$  (from the policy improvement theorem)

- Monotonic  $\epsilon$ -greedy policy improvement theorem is a sanity check about using  $\epsilon$ -greedy policies for policy iteration
- But note the assumption when we learned about policy iteration was that policy evaluation was done **exactly**
- In the last lecture we learned about doing policy evaluation without access to the true dynamics and reward models
- MC and TD yielded estimates of  $V^\pi$
- Should not yet be clear that we can ensure monotonic improvement or convergence given such estimates...

# Break



# Today: Model-free Control

- Generalized policy improvement
- Importance of exploration
- **Monte Carlo control**
- Model-free control with temporal difference (SARSA, Q-learning)
- Maximization bias

# Recall Monte Carlo Policy Evaluation, Now for Q

---

```
1: Initialize  $Q(s, a) = 0, N(s, a) = 0 \forall (s, a), k = 1$ , Input  $\epsilon = 1, \pi$ 
2: loop
3:   Sample  $k$ -th episode  $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \dots, s_{k,T})$  given  $\pi$ 
3:   Compute  $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \dots \gamma^{T_i-1} r_{k,T_i} \forall t$ 
4:   for  $t = 1, \dots, T$  do
5:     if First visit to  $(s,a)$  in episode  $k$  then
6:        $N(s, a) = N(s, a) + 1$ 
7:        $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$ 
8:     end if
9:   end for
10:   $k = k + 1$ 
11: end loop
```

---

# Monte Carlo Online Control / On Policy Improvement

---

```
1: Initialize  $Q(s, a) = 0, N(s, a) = 0 \forall (s, a)$ , Set  $\epsilon = 1, k = 1$ 
2:  $\pi_k = \epsilon$ -greedy( $Q$ ) // Create initial  $\epsilon$ -greedy policy
3: loop
4:   Sample  $k$ -th episode  $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \dots, s_{k,T})$  given  $\pi_k$ 
4:    $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \dots + \gamma^{T-t} r_{k,T}$ 
5:   for  $t = 1, \dots, T$  do
6:     if First visit to  $(s, a)$  in episode  $k$  then
7:        $N(s, a) = N(s, a) + 1$ 
8:        $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$ 
9:     end if
10:  end for
11:   $k = k + 1, \epsilon = 1/k$ 
12:   $\pi_k = \epsilon$ -greedy( $Q$ ) // Policy improvement
13: end loop
```

---

# Poll. Check Your Understanding: MC for On Policy Control

- Mars rover with new actions:
  - $r(-, a_1) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 +10]$ ,  $r(-, a_2) = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 +5]$ ,  $\gamma = 1$ .
- Assume current greedy  $\pi(s) = a_1 \ \forall s$ ,  $\epsilon = .5$ .  $Q(s, a) = 0$  for all  $(s, a)$
- Sample trajectory from  $\epsilon$ -greedy policy
- Trajectory =  $(s_3, a_1, 0, s_2, a_2, 0, s_3, a_1, 0, s_2, a_2, 0, s_1, a_1, 1, \text{terminal})$
- First visit MC estimate of  $Q$  of each  $(s, a)$  pair?
- $Q^{\epsilon-\pi}(-, a_1) = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]$

After this trajectory (Select all)

- $Q^{\epsilon-\pi}(-, a_2) = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$
- The new **greedy** policy would be:  $\pi = [1 \ \text{tie} \ 1 \ \text{tie} \ \text{tie} \ \text{tie} \ \text{tie}]$
- The new **greedy** policy would be:  $\pi = [1 \ 2 \ 1 \ \text{tie} \ \text{tie} \ \text{tie} \ \text{tie}]$
- If  $\epsilon = 1/3$ , prob of selecting  $a_1$  in  $s_1$  in the new  $\epsilon$ -greedy policy is  $1/9$ .
- If  $\epsilon = 1/3$ , prob of selecting  $a_1$  in  $s_1$  in the new  $\epsilon$ -greedy policy is  $2/3$ .
- If  $\epsilon = 1/3$ , prob of selecting  $a_1$  in  $s_1$  in the new  $\epsilon$ -greedy policy is  $5/6$ .
- Not sure

# Check Your Understanding: MC for On Policy Control

- Mars rover with new actions:
  - $r(-, a_1) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ +10]$ ,  $r(-, a_2) = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ +5]$ ,  $\gamma = 1$ .
- Assume current greedy  $\pi(s) = a_1 \ \forall s$ ,  $\epsilon = .5$
- Sample trajectory from  $\epsilon$ -greedy policy
- Trajectory =  $(s_3, a_1, 0, s_2, a_2, 0, s_3, a_1, 0, s_2, a_2, 0, s_1, a_1, 1, \text{terminal})$
- First visit MC estimate of  $Q$  of each  $(s, a)$  pair?
- $Q^{\epsilon-\pi}(-, a_1) = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]$ ,  $Q^{\epsilon-\pi}(-, a_2) = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]$
- What is  $\pi(s) = \arg \max_a Q^{\epsilon-\pi}(s, a) \ \forall s$ ?  
 $\pi = [1 \ 2 \ 1 \ \text{tie} \ \text{tie} \ \text{tie} \ \text{tie}]$
- Under the new  $\epsilon$ -greedy policy, if  $k = 3$ ,  $\epsilon = 1/k$   
With probability  $2/3$  choose  $\pi(s)$  else choose randomly. As an example,  $\pi(s_1) = a_1$  with prob  $(2/3)$  else randomly choose an action.  
So the prob of picking  $a_1$  will be  $2/3 + (1/3) * (1/2) = 5/6$

# MC control with $\epsilon$ -greedy policies

- Is the prior algorithm a reasonable one?
- Will it eventually converge to the optimal  $Q^*$  function?
- Now see a set of conditions that is sufficient to ensure this desirable outcome

# Greedy in the Limit of Infinite Exploration (GLIE)

## Definition of GLIE

- All state-action pairs are visited an infinite number of times

$$\lim_{i \rightarrow \infty} N_i(s, a) \rightarrow \infty$$

- Behavior policy (policy used to act in the world) converges to greedy policy

$$\lim_{i \rightarrow \infty} \pi(a|s) \rightarrow \arg \max_a Q(s, a) \text{ with probability 1}$$

# Greedy in the Limit of Infinite Exploration (GLIE)

## Definition of GLIE

- All state-action pairs are visited an infinite number of times

$$\lim_{i \rightarrow \infty} N_i(s, a) \rightarrow \infty$$

- Behavior policy (policy used to act in the world) converges to greedy policy

$$\lim_{i \rightarrow \infty} \pi(a|s) \rightarrow \arg \max_a Q(s, a) \text{ with probability 1}$$

- A simple GLIE strategy is  $\epsilon$ -greedy where  $\epsilon$  is reduced to 0 with the following rate:  $\epsilon_i = 1/i$



## Theorem

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

# Break

# Today: Model-free Control

- Generalized policy improvement
- Importance of exploration
- Monte Carlo control
- **Model-free control with temporal difference (SARSA, Q-learning)**
- Maximization bias

# Model-free Policy Iteration with TD Methods

- Use temporal difference methods for policy evaluation step
- Initialize policy  $\pi$
- Repeat:
  - Policy evaluation: compute  $Q^\pi$  using temporal difference updating with  $\epsilon$ -greedy policy
  - Policy improvement: Same as Monte carlo policy improvement, set  $\pi$  to  $\epsilon$ -greedy ( $Q^\pi$ )
- Important issue: is it necessary for policy evaluation to converge to the true value of  $Q^\pi$  before updating the policy?
- Answer: Perhaps surprisingly, no. A single update of TD (policy evaluation updating of  $Q$ ) can be sufficient!

# Model-free Policy Iteration with TD Methods

- Use temporal difference methods for policy evaluation step
- Initialize policy  $\pi$
- Repeat:
  - Policy evaluation: compute  $Q^\pi$  using temporal difference updating with  $\epsilon$ -greedy policy
  - Policy improvement: Same as Monte carlo policy improvement, set  $\pi$  to  $\epsilon$ -greedy ( $Q^\pi$ )
- First consider SARSA, which is an on-policy algorithm.
- On policy: SARSA is trying to compute an estimate  $Q$  of the policy being followed.

# General Form of SARSA Algorithm

- 
- 1: Set initial  $\epsilon$ -greedy policy  $\pi$  randomly,  $t = 0$ , initial state  $s_t = s_0$
  - 2: Take  $a_t \sim \pi(s_t)$
  - 3: Observe  $(r_t, s_{t+1})$
  - 4: **loop**
  - 5:   Take action  $a_{t+1} \sim \pi(s_{t+1})$  // Sample action from policy
  - 6:   Observe  $(r_{t+1}, s_{t+2})$
  - 7:   Update Q given  $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ :
  - 8:   Perform policy improvement:
  - 9:    $t = t + 1$
  - 10: **end loop**
-

# General Form of SARSA Algorithm

- 
- 1: Set initial  $\epsilon$ -greedy policy  $\pi$ ,  $t = 0$ , initial state  $s_t = s_0$
  - 2: Take  $a_t \sim \pi(s_t)$  // Sample action from policy
  - 3: Observe  $(r_t, s_{t+1})$
  - 4: **loop**
  - 5:   Take action  $a_{t+1} \sim \pi(s_{t+1})$
  - 6:   Observe  $(r_{t+1}, s_{t+2})$
  - 7:    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  - 8:    $\pi(s_t) = \arg \max_a Q(s_t, a)$  w.prob  $1 - \epsilon$ , else random
  - 9:    $t = t + 1$
  - 10: **end loop**
-

# Worked Example: SARSA for Mars Rover

- 
- 1: Set initial  $\epsilon$ -greedy policy  $\pi$ ,  $t = 0$ , initial state  $s_t = s_0$
  - 2: Take  $a_t \sim \pi(s_t)$  // Sample action from policy
  - 3: Observe  $(r_t, s_{t+1})$
  - 4: **loop**
  - 5:   Take action  $a_{t+1} \sim \pi(s_{t+1})$
  - 6:   Observe  $(r_{t+1}, s_{t+2})$
  - 7:    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  - 8:    $\pi(s_t) = \arg \max_a Q(s_t, a)$  w.prob  $1 - \epsilon$ , else random
  - 9:    $t = t + 1$
  - 10: **end loop**
- 
- Initialize  $\epsilon = 1/k$ ,  $k = 1$ , and  $\alpha = 0.5$ ,  $Q(-, a_1) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +10]$ ,  $Q(-, a_2) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +5]$ ,  $\gamma = 1$
  - Assume starting state is  $s_6$  and sample  $a_1$



# Worked Example: SARSA for Mars Rover

- 
- 1: Set initial  $\epsilon$ -greedy policy  $\pi$ ,  $t = 0$ , initial state  $s_t = s_0$
  - 2: Take  $a_t \sim \pi(s_t)$  // Sample action from policy
  - 3: Observe  $(r_t, s_{t+1})$
  - 4: **loop**
  - 5:   Take action  $a_{t+1} \sim \pi(s_{t+1})$
  - 6:   Observe  $(r_{t+1}, s_{t+2})$
  - 7:    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  - 8:    $\pi(s_t) = \arg \max_a Q(s_t, a)$  w.prob  $1 - \epsilon$ , else random
  - 9:    $t = t + 1$
  - 10: **end loop**
- 

- Initialize  $\epsilon = 1/k$ ,  $k = 1$ , and  $\alpha = 0.5$ ,  $Q(-, a_1) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +10]$ ,  $Q(-, a_2) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +5]$ ,  $\gamma = 1$
- Tuple:  $(s_6, a_1, 0, s_7, a_2, 5, s_7)$ .
- $Q(s_6, a_1) = .5 * 0 + .5 * (0 + \gamma Q(s_7, a_2)) = 2.5$

# SARSA Initialization

- Mars rover with new actions:
  - $r(-, a_1) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 + 10]$ ,  $r(-, a_2) = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 + 5]$ ,  $\gamma = 1$ .
- Initialize  $\epsilon = 1/k$ ,  $k = 1$ , and  $\alpha = 0.5$ ,  $Q(-, a_1) = r(-, a_1)$ ,  
 $Q(-, a_2) = r(-, a_2)$
- SARSA:  $(s_6, a_1, 0, s_7, a_2, 5, s_7)$ .
- Does how  $Q$  is initialized matter (initially? asymptotically)?  
Asymptotically no, under mild conditions, but at the beginning, yes

# Convergence Properties of SARSA

## Theorem

SARSA for finite-state and finite-action MDPs converges to the optimal action-value,  $Q(s, a) \rightarrow Q^*(s, a)$ , under the following conditions:

- 1 The policy sequence  $\pi_t(a|s)$  satisfies the condition of GLIE
- 2 The step-sizes  $\alpha_t$  satisfy the Robbins-Munro sequence such that

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

- For ex.  $\alpha_t = \frac{1}{t}$  satisfies the above condition.

# Q-Learning: Learning the Optimal State-Action Value

- SARSA is an **on-policy** learning algorithm
- SARSA estimates the value of the current **behavior** policy (policy using to take actions in the world)
- And then updates the policy trying to estimate
- Alternatively, can we directly estimate the value of  $\pi^*$  while acting with another behavior policy  $\pi_b$ ?
- Yes! Q-learning, an **off-policy** RL algorithm

# On and Off-Policy Learning

- On-policy learning
  - Direct experience
  - Learn to estimate and evaluate a policy from experience obtained from following that policy
- Off-policy learning
  - Learn to estimate and evaluate a policy using experience gathered from following a different policy

# Q-Learning: Learning the Optimal State-Action Value

- SARSA is an **on-policy** learning algorithm
- SARSA estimates the value of the current **behavior** policy (policy using to take actions in the world)
- And then updates the policy trying to estimate
- Alternatively, can we directly estimate the value of  $\pi^*$  while acting with another behavior policy  $\pi_b$ ?
- Yes! Q-learning, an **off-policy** RL algorithm
- Key idea: Maintain state-action  $Q$  estimates and use to bootstrap—use the value of the best future action
- Recall SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t))$$

- Q-learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t))$$

# Q-Learning with $\epsilon$ -greedy Exploration

- 
- 1: Initialize  $Q(s, a), \forall s \in S, a \in A$   $t = 0$ , initial state  $s_t = s_0$
  - 2: Set  $\pi_b$  to be  $\epsilon$ -greedy w.r.t.  $Q$
  - 3: **loop**
  - 4:   Take  $a_t \sim \pi_b(s_t)$  // Sample action from policy
  - 5:   Observe  $(r_t, s_{t+1})$
  - 6:    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
  - 7:    $\pi(s_t) = \arg \max_a Q(s_t, a)$  w.prob  $1 - \epsilon$ , else random
  - 8:    $t = t + 1$
  - 9: **end loop**
-

# Worked Example: $\epsilon$ -greedy Q-Learning Mars

- 
- 1: Initialize  $Q(s, a), \forall s \in S, a \in A$   $t = 0$ , initial state  $s_t = s_0$
  - 2: Set  $\pi_b$  to be  $\epsilon$ -greedy w.r.t.  $Q$
  - 3: **loop**
  - 4:   Take  $a_t \sim \pi_b(s_t)$  // Sample action from policy
  - 5:   Observe  $(r_t, s_{t+1})$
  - 6:    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
  - 7:    $\pi(s_t) = \arg \max_a Q(s_t, a)$  w.prob  $1 - \epsilon$ , else random
  - 8:    $t = t + 1$
  - 9: **end loop**
- 

- Initialize  $\epsilon = 1/k$ ,  $k = 1$ , and  $\alpha = 0.5$ ,  $Q(-, a_1) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +10]$ ,  $Q(-, a_2) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +5]$ ,  $\gamma = 1$
- Like in SARSA example, start in  $s_6$  and take  $a_1$ .



# Worked Example: $\epsilon$ -greedy Q-Learning Mars

- 
- 1: Initialize  $Q(s, a), \forall s \in S, a \in A$   $t = 0$ , initial state  $s_t = s_0$
  - 2: Set  $\pi_b$  to be  $\epsilon$ -greedy w.r.t.  $Q$
  - 3: **loop**
  - 4:   Take  $a_t \sim \pi_b(s_t)$  // Sample action from policy
  - 5:   Observe  $(r_t, s_{t+1})$
  - 6:    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
  - 7:    $\pi(s_t) = \arg \max_a Q(s_t, a)$  w.prob  $1 - \epsilon$ , else random
  - 8:    $t = t + 1$
  - 9: **end loop**
- 

- Initialize  $\epsilon = 1/k$ ,  $k = 1$ , and  $\alpha = 0.5$ ,  $Q(-, a_1) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +10]$ ,  $Q(-, a_2) = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ +5]$ ,  $\gamma = 1$
- Tuple:  $(s_6, a_1, 0, s_7)$ .
- $Q(s_6, a_1) = 0 + .5 * (0 + \gamma \max_{a'} Q(s_7, a') - 0) = .5 * 10 = 5$
- Recall that in the SARSA update we saw  $Q(s_6, a_1) = 2.5$  because we used the actual action taken at  $s_7$  instead of the max
- Does how  $Q$  is initialized matter (initially? asymptotically?)?  
Asymptotically no, under mild conditions, but at the beginning, yes

# Check Your Understanding: SARSA and Q-Learning

- SARSA:  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
- Q-Learning:  
 $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

Select all that are true

- 1 Both SARSA and Q-learning may update their policy after every step
- 2 If  $\epsilon = 0$  for all time steps, and  $Q$  is initialized randomly, a SARSA  $Q$  state update will be the same as a Q-learning  $Q$  state update
- 3 Not sure

# Check Your Understanding: SARSA and Q-Learning

- SARSA:  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
- Q-Learning:  
 $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

Select all that are true

- 1 Both SARSA and Q-learning may update their policy after every step
- 2 If  $\epsilon = 0$  for all time steps, and  $Q$  is initialized randomly, a SARSA  $Q$  state update will be the same as a Q-learning  $Q$  state update
- 3 Not sure

Both are true.

# Q-Learning with $\epsilon$ -greedy Exploration

- What conditions are sufficient to ensure that Q-learning with  $\epsilon$ -greedy exploration converges to optimal  $Q^*$ ?

Visit all  $(s, a)$  pairs infinitely often, and the step-sizes  $\alpha_t$  satisfy the Robbins-Munro sequence. Note: the algorithm does not have to be greedy in the limit of infinite exploration (GLIE) to satisfy this (could keep  $\epsilon$  large).

- What conditions are sufficient to ensure that Q-learning with  $\epsilon$ -greedy exploration converges to optimal  $\pi^*$ ?

The algorithm is GLIE, along with the above requirement to ensure the Q value estimates converge to the optimal Q.

# Break

# Maximization Bias<sup>1</sup>

- Consider single-state MDP ( $|S| = 1$ ) with 2 actions, and both actions have 0-mean **random** rewards, ( $\mathbb{E}(r|a = a_1) = \mathbb{E}(r|a = a_2) = 0$ ).
- Then  $Q(s, a_1) = Q(s, a_2) = 0 = V(s)$
- Assume there are prior samples of taking action  $a_1$  and  $a_2$
- Let  $\hat{Q}(s, a_1), \hat{Q}(s, a_2)$  be the finite sample estimate of  $Q$
- Use an unbiased estimator for  $Q$ : e.g.  $\hat{Q}(s, a_1) = \frac{1}{n(s, a_1)} \sum_{i=1}^{n(s, a_1)} r_i(s, a_1)$
- Let  $\hat{\pi} = \arg \max_a \hat{Q}(s, a)$  be the greedy policy w.r.t. the estimated  $\hat{Q}$

---

<sup>1</sup>Example from Mannor, Simester, Sun and Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. Management Science 2007

# Maximization Bias<sup>2</sup> Proof

- Consider single-state MDP ( $|S| = 1$ ) with 2 actions, and both actions have 0-mean random rewards, ( $\mathbb{E}(r|a = a_1) = \mathbb{E}(r|a = a_2) = 0$ ).
- Then  $Q(s, a_1) = Q(s, a_2) = 0 = V(s)$
- Assume there are prior samples of taking action  $a_1$  and  $a_2$
- Let  $\hat{Q}(s, a_1), \hat{Q}(s, a_2)$  be the finite sample estimate of  $Q$
- Use an unbiased estimator for  $Q$ : e.g.  $\hat{Q}(s, a_1) = \frac{1}{n(s, a_1)} \sum_{i=1}^{n(s, a_1)} r_i(s, a_1)$
- Let  $\hat{\pi} = \arg \max_a \hat{Q}(s, a)$  be the greedy policy w.r.t. the estimated  $\hat{Q}$
- *Even though each estimate of the state-action values is unbiased, the estimate of  $\hat{\pi}$ 's value  $\hat{V}^{\hat{\pi}}$  can be biased:*
$$\begin{aligned}\hat{V}^{\hat{\pi}}(s) &= \mathbb{E}[\max \hat{Q}(s, a_1), \hat{Q}(s, a_2)] \\ &\geq \max[\mathbb{E}[\hat{Q}(s, a_1)], [\hat{Q}(s, a_2)]] \\ &= \max[0, 0] = V^{\pi},\end{aligned}$$
where the inequality comes from Jensen's inequality.

---

<sup>2</sup>Example from Mannor, Simester, Sun and Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. Management Science 2007

# Double Q-Learning

- The greedy policy w.r.t. estimated  $Q$  values can yield a maximization bias during finite-sample learning
- Avoid using max of estimates as estimate of max of true values
- Instead split samples and use to create two independent unbiased estimates of  $Q_1(s_1, a_i)$  and  $Q_2(s_1, a_i) \forall a$ .
  - Use one estimate to select max action:  $a^* = \arg \max_a Q_1(s_1, a)$
  - Use other estimate to estimate value of  $a^*$ :  $Q_2(s, a^*)$
  - Yields unbiased estimate:  $\mathbb{E}(Q_2(s, a^*)) = Q(s, a^*)$



# Double Q-Learning

- The greedy policy w.r.t. estimated  $Q$  values can yield a maximization bias during finite-sample learning
- Avoid using max of estimates as estimate of max of true values
- Instead split samples and use to create two independent unbiased estimates of  $Q_1(s_1, a_i)$  and  $Q_2(s_1, a_i) \forall a$ .
  - Use one estimate to select max action:  $a^* = \arg \max_a Q_1(s_1, a)$
  - Use other estimate to estimate value of  $a^*$ :  $Q_2(s, a^*)$
  - Yields unbiased estimate:  $\mathbb{E}(Q_2(s, a^*)) = Q(s, a^*)$
- Why does this yield an unbiased estimate of the max state-action value?

Using independent samples to estimate the value
- If acting online, can alternate samples used to update  $Q_1$  and  $Q_2$ , using the other to select the action chosen
- Next slides extend to full MDP case (with more than 1 state)

# Double Q-Learning

---

```
1: Initialize  $Q_1(s, a)$  and  $Q_2(s, a), \forall s \in S, a \in A$   $t = 0$ , initial state  $s_t = s_0$ 
2: loop
3:   Select  $a_t$  using  $\epsilon$ -greedy  $\pi(s) = \arg \max_a Q_1(s_t, a) + Q_2(s_t, a)$ 
4:   Observe  $(r_t, s_{t+1})$ 
5:   if (with 0.5 probability) then
6:      $Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, \arg \max_a Q_1(s_{t+1}, a)) - Q_1(s_t, a_t))$ 
7:   else
8:      $Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, \arg \max_a Q_2(s_{t+1}, a)) - Q_2(s_t, a_t))$ 
9:   end if
10:   $t = t + 1$ 
11: end loop
```

---

Compared to Q-learning, how does this change the: memory requirements, computation requirements per step, amount of data required?

# Double Q-Learning

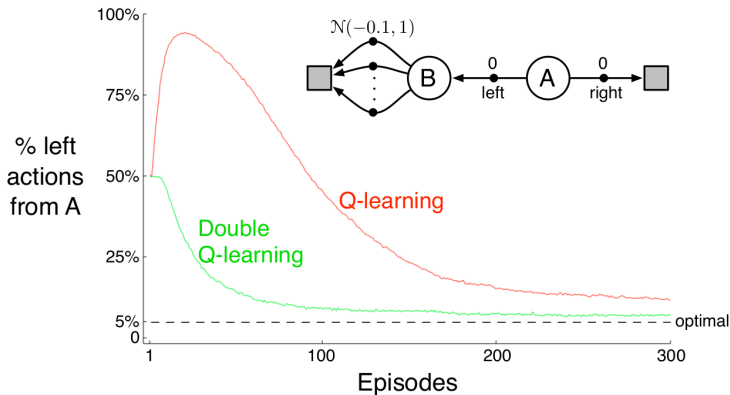
---

```
1: Initialize  $Q_1(s, a)$  and  $Q_2(s, a), \forall s \in S, a \in A$   $t = 0$ , initial state  $s_t = s_0$ 
2: loop
3:   Select  $a_t$  using  $\epsilon$ -greedy  $\pi(s) = \arg \max_a Q_1(s_t, a) + Q_2(s_t, a)$ 
4:   Observe  $(r_t, s_{t+1})$ 
5:   if (with 0.5 probability) then
6:      $Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, \arg \max_a Q_1(s_{t+1}, a)) - Q_1(s_t, a_t))$ 
7:   else
8:      $Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, \arg \max_a Q_2(s_{t+1}, a)) - Q_2(s_t, a_t))$ 
9:   end if
10:   $t = t + 1$ 
11: end loop
```

---

Compared to Q-learning, how does this change the: memory requirements, computation requirements per step, amount of data required?

# Double Q-Learning (Figure 6.7 in Sutton and Barto 2018)



Due to the maximization bias, Q-learning spends much more time selecting suboptimal actions than double Q-learning.

# What You Should Know

- Be able to implement MC on policy control and SARSA and Q-learning
- Compare them according to properties of how quickly they update, (informally) bias and variance, computational cost
- Define conditions for these algorithms to converge to the optimal Q and optimal  $\pi$  and give at least one way to guarantee such conditions are met.

# Class Structure

- Last time: Policy evaluation with no knowledge of how the world works (MDP model not given)
- This time: Control (making decisions) without a model of how the world works
- **Next time: Generalization – Value function approximation**

# Today: Learning to Control Involves

- Optimization: Goal is to identify a policy with high expected rewards (similar to Lecture 2 on computing an optimal policy given decision process models)
- Delayed consequences: May take many time steps to evaluate whether an earlier decision was good or not
- Exploration: Necessary to try different actions to learn what actions can lead to high rewards

# Recall: Reinforcement Learning Involves

- Optimization
- Delayed consequences
- Exploration
- Generalization



# Evaluation to Control

- Last time: how good is a specific policy?
  - Given no access to the decision process model parameters
  - Instead have to estimate from data / experience
- Today: how can we learn a good policy?

# Evaluation to Control

- Last time: how good is a specific policy?
  - Given no access to the decision process model parameters
  - Instead have to estimate from data / experience
- Today: how can we learn a good policy?