Lecture 4: Model Free Control

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2022

- 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 L4. Polleverywhere 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

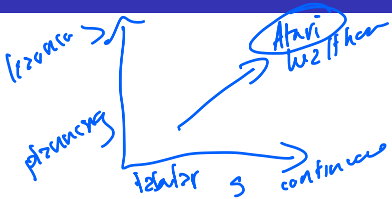
Refresh Your Knowledge L4. Polleverywhere Poll

- 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

- OAE please email us asap @ staff
  mailing list
- HW1 6pm Stanford time Friday

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

Follow Ups from Last Time: Step Sizes for TD Learning

- In TD learning the value of the current state under the policy π is updated as

$$V^\pi(s_t) = V^\pi(s_t) + \alpha_t(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$$

$$= (1-\alpha_t) V^\pi(s_t) + \alpha_t (r_t + \gamma V^\pi(s_{t+1}))$$

- Note in last class we simply used α and here we are using α_t to indicate that it could change over time
- Assume the step-sizes α_t satisfy a Robbins-Munro sequence

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

$$r_t + \gamma V^\pi(s_{t+1})$$
$$r_t + \gamma r_{t+1} + \gamma^2 V^\pi(s_{t+2})$$

- Then TD learning converges with probability 1 to $V^\pi(s)$ for each state, as long as each state is visited infinitely often under the policy π.
- Note $\alpha_t = \frac{1}{T}$ satisfies the above condition.
- Sidenote: why TD(0)? There are variants of TD that extrapolate between the TD learning variant from last class, to using longer returns as targets as in MC. See Sutton and Barto for more details.

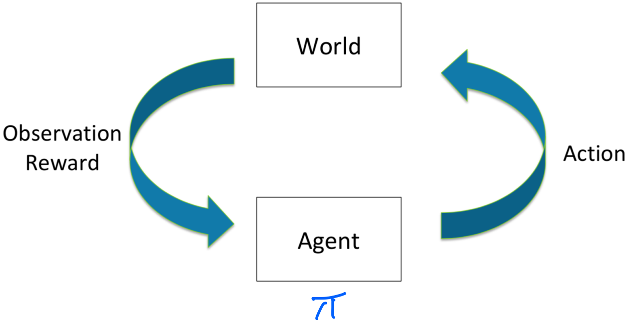
Today: Model-free Control

- 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 locomation, 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

tabular

π^*

- Initialize policy π
- Repeat:
  - Policy evaluation: compute V^π (lecture 3)
  - Policy improvement: update π

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

- 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

$\pi \rightarrow V^{\pi}$

$\pi \rightarrow Q^{\pi}$

- Initialize policy π
- Repeat:
  - Policy evaluation: compute $\boxed{Q^{\pi}}$
  - Policy improvement: update π

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 L4N1: 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 π_{i+1} deterministic or stochastic?
- Answer: Deterministic, Stochastic, Not Sure
- Recall in model-free policy evaluation, we estimated V^π by using π to generate new trajectories
- Question: Can we compute $Q^{\pi_{i+1}}(s,a)\ \forall s, a$ by using this π_{i+1} to generate new trajectories?
- Answer: True, False, Not Sure

$s, \pi_{i+1}(s)$

$a \neq \pi_{i+1}(s)$

Check Your Understanding L4N1: Model-free Generalized Policy Improvement

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

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

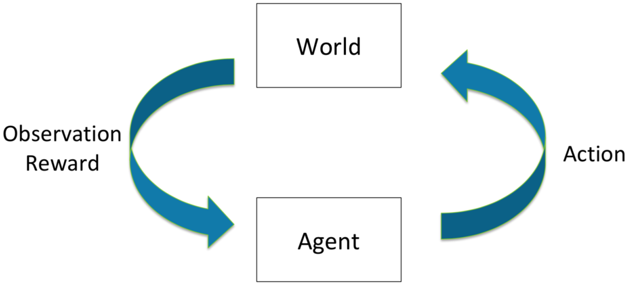
- Question: is this π_{i+1} deterministic or stochastic?
- :Answer: Deterministic
- Recall in model-free policy evaluation, we estimated V^{π} by using π to generate new trajectories
- Question: Can we compute $Q^{\pi_{i+1}}(s,a)\ \forall s,a$ by using this π_{i+1} to generate new trajectories?
- Answer: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

Model-free Policy Iteration

- Initialize policy π
- Repeat:
  - Policy evaluation: compute Q^π
  - Policy improvement: update π given Q^π

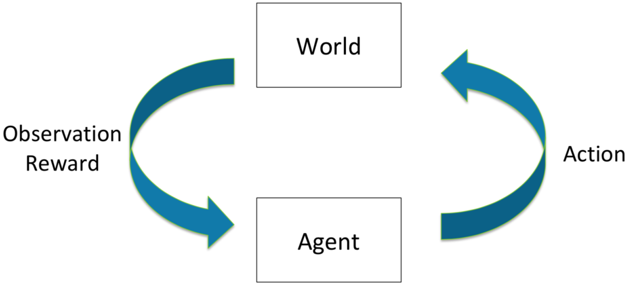
- May need to modify policy evaluation:
  - If π is deterministic, can't compute $Q(s, a)$ for any $a \neq \pi(s)$
- How to interleave policy evaluation and improvement?
  - Policy improvement is now using an estimated Q

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^π
- In general seems subtle
  - Need to try all (s, a) pairs but then follow π
  - Want to ensure resulting estimate Q^π 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^π converges to the true value

ϵ-greedy Policies

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

with prob $1 - \epsilon$ $\pi(s) = \arg\max_a Q(s, a)$

"greedy"

$\frac{\epsilon}{|A|}$ action a_i

ϵ-greedy Policies

- Simple idea to balance exploration and exploitation
- Let $|A|$ be the number of actions
- Then an ϵ-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 $\frac{\epsilon}{|A|}$

Policy Improvement with ϵ-greedy policies

- We will now prove an important property
- Before doing so, it is helpful to remember that $V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(a,s)$

det policy $\pi(a|s) = 1$ or 0

$V^\pi(s) = Q^\pi(\pi(s), s)$

Monotonic ϵ-greedy Policy Improvement

Theorem

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

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

definition

$\pi_{i+1}(a|s)$

$$= \frac{\epsilon}{|A|} \sum_a Q^{\pi_i}(s,a) + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \left[\frac{1-\epsilon}{1-\epsilon}\right]$$

$$\left[\frac{1}{(1-\epsilon)} \sum_a \left(\pi_i(a|s) - \epsilon/|A|\right)\right]$$

$$\geq \quad (1-\epsilon) \frac{1}{(1-\epsilon)} \sum_a Q^{\pi_i}(s,a) \left(\pi_i(a|s) - \frac{\epsilon}{|A|}\right)$$

$$= \sum_a Q^{\pi_i}(s,a) \pi_i(a|s)$$

$$= V^{\pi_i}(s)$$

Monotonic ϵ-greedy Policy Improvement

Theorem

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

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

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

defn π_i

$\sum_a \max_a Q^{\pi_i}(s,a)$

$\geq \sum_a Q^{\pi_i}(s,a)$

policy improvement $\quad V^{\pi_{i+1}} \geq V^{\pi}$

ϵ-greedy

- Monotonic ϵ-greedy policy improvement theorem is a sanity check about using ϵ-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^π
- Should not yet be clear that we can ensure monotonic improvement or convergence given such estimates...

$Q^{\pi_i}(s,a)$

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

Recall Monte Carlo Policy Evaluation, **Now for Q**

1: Initialize $Q(s,a) = 0, N(s,a) = 0 \ \forall (s,a)$, $k = 1$, Input $\epsilon = 1$, π
2: **loop**
3: Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π
3: Compute $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \cdots \gamma^{T_i - 1} r_{k,T_i} \ \forall t$
4: **for** $t = 1, \ldots, T$ **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**

Initialize $Q(s, a) = 0, N(s, a) = 0 \ \forall(s, a)$, Set $\epsilon = 1, \ k = 1$
$\pi_k = \epsilon$-greedy(Q) // Create initial ϵ-greedy policy
loop
  Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π_k
  $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \cdots \gamma^{T_i - 1} r_{k,T_i}$
  for $t = 1, \ldots, T$ **do**
    if First visit to (s, a) in episode k **then**
      $N(s, a) = N(s, a) + 1$
      $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
    end if
  end for
  $k = k + 1$, $\boxed{\epsilon = 1/k}$
  $\pi_k = \epsilon$-greedy(Q) // Policy improvement
end loop

Poll. Check Your Understanding L4N2: MC for On Policy Control

- Mars rover with new actions:
  - $r(-, a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $r(-, a_2) = [\ 0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$.
- Assume current greedy $\pi(s) = a_1\ \forall s$, ϵ=.5. $Q(s, a) = 0$ for all (s, a)
- Sample trajectory from ϵ-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: $\pi = [1$ tie 1 tie tie tie tie$]$
- The new **greedy** policy would be: $\pi = [1\ 2\ 1$ tie tie tie tie$]$
- If $\epsilon = 1/3$, prob of selecting a_1 in s_1 in the new ϵ-greedy policy is $1/9$.
- If $\epsilon = 1/3$, prob of selecting a_1 in s_1 in the new ϵ-greedy policy is $2/3$.
- If $\epsilon = 1/3$, prob of selecting a_1 in s_1 in the new ϵ-greedy policy is $5/6$.
- Not sure

Check Your Understanding L4N2: MC for On Policy Control

- Mars rover with new actions:
  - $r(-, a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $r(-, a_2) = [\ 0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$.
- Assume current greedy $\pi(s) = a_1\ \forall s$, ϵ=.5
- Sample trajectory from ϵ-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, 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 $\pi(s) = \arg\max_a Q^{\epsilon-\pi}(s, a)\ \forall s$?
  π = [1 2 1 tie tie tie tie]

  greedy

  $\epsilon = 1/3$
- Under the new ϵ-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 $\underbrace{2/3}_{} + \underbrace{(1/3)}_{} * \underbrace{(1/2)}_{} = 5/6$

MC control with ϵ-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

$$\lim_{i \to \infty} N_i(s, a) \to \infty \quad \forall s, a$$

\# total decisions (t)

- Behavior policy (policy used to act in the world) converges to greedy policy
  $\lim_{i \to \infty} \pi(a|s) \to \arg\max_a Q(s, a)$ 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

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

- Behavior policy (policy used to act in the world) converges to greedy policy
  $\lim_{i \to \infty} \pi(a|s) \to \arg\max_a Q(s,a)$ with probability 1

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

i = # decisions / episodes

GLIE Monte-Carlo Control

Theorem

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

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 π
- Repeat:
  - Policy evaluation: compute Q^π using temporal difference updating with ϵ-greedy policy
  - Policy improvement: Same as Monte carlo policy improvement, set π to ϵ-greedy (Q^π)
- Important issue: is it necessary for policy evaluation to converge to the true value of Q^π before updating the policy?
- Answer: Perhaps surprizingly, 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 π
- Repeat:
  - Policy evaluation: compute Q^π using temporal difference updating with ϵ-greedy policy
  - Policy improvement: Same as Monte carlo policy improvement, set π to ϵ-greedy (Q^π)
- 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

```
Set initial ε-greedy policy π randomly, t = 0, initial state s_t = s_0
Take a_t ~ π(s_t)
Observe (r_t, s_{t+1})
loop
    Take action a_{t+1} ~ π(s_{t+1}) // Sample action from policy
    Observe (r_{t+1}, s_{t+2})
    Update Q given (s_t, a_t, r_t, s_{t+1}, a_{t+1}):


    Perform policy improvement:


    t = t + 1
end loop
```

General Form of SARSA Algorithm

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$

Take $a_t \sim \pi(s_t)$ // Sample action from policy

Observe (r_t, s_{t+1})

loop

  Take action $a_{t+1} \sim \pi(s_{t+1})$

  Observe (r_{t+1}, s_{t+2})

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

  $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random

  $t = t + 1$

end loop

TD $r_t + \gamma V^{\pi}(s_{t+1})$

policy improvement

Worked Example: SARSA for Mars Rover

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$
Take $a_t \sim \pi(s_t)$ // Sample action from policy
Observe (r_t, s_{t+1})
loop
    Take action $a_{t+1} \sim \pi(s_{t+1})$
    Observe (r_{t+1}, s_{t+2})
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
    $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
    $t = t + 1$
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

$(s_6, a_1, 0, s_7, a_2, 5\ \text{term})$

$Q(s_6, a_1) = .5\ Q(s_6, a_1) + .5(0 + \gamma Q(s_7, a_2))$
$= 2.5$

Worked Example: SARSA for Mars Rover

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$
Take $a_t \sim \pi(s_t)$ // Sample action from policy
Observe (r_t, s_{t+1})
loop
    Take action $a_{t+1} \sim \pi(s_{t+1})$
    Observe (r_{t+1}, s_{t+2}) π
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
    $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
    $t = t + 1$
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:
  - $r(-, a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $r(-, a_2) = [\ 0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$.
- 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 condiditions, but at the beginning, yes

Convergence Properties of SARSA

Theorem

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

same as MC

1. The policy sequence $\pi_t(a|s)$ satisfies the condition of GLIE
2. The step-sizes α_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.

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 that (behavior) policy
- Alternatively, can we directly estimate the value of π^* while acting with another behavior policy π_b?
- Yes! Q-learning, an **off-policy** RL algorithm

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 π^* while acting with another behavior policy π_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 ϵ-greedy Exploration

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

Worked Example: ϵ-greedy Q-Learning Mars

```
Initialize Q(s,a), ∀s ∈ S, a ∈ A  t = 0, initial state s_t = s_0
Set π_b to be ε-greedy w.r.t. Q
loop
    Take a_t ~ π_b(s_t) // Sample action from policy
    Observe (r_t, s_{t+1})
    Q(s_t, a_t) ← Q(s_t, a_t) + α(r_t + γ max_a Q(s_{t+1}, a) − Q(s_t, a_t))
    π(s_t) = arg max_a Q(s_t, a) w.prob 1 − ε, else random
    t = t + 1
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: ϵ-greedy Q-Learning Mars

Initialize $Q(s,a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
Set π_b to be ϵ-greedy w.r.t. Q
loop
  Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
  Observe (r_t, s_{t+1})
  $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t))$
  $\pi(s_t) = \arg\max_a Q(s_t,a)$ w.prob $1-\epsilon$, else random
  $t = t + 1$
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 condiditions, but at the beginning, yes

Check Your Understanding L4N3: 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 L4N3: 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 ϵ-greedy Exploration

- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal Q^*?
  Visit all (s, a) pairs infinitely often, and the step-sizes α_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 ϵ large).

- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal π^*?
  The algorithm is GLIE, along with the above requirement to ensure the Q value estimates converge to the optimal Q.

Break

Maximization Bias[1]

- 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 $\hat{Q}$

$H = 1$

$$\hat{V}^{\hat{\pi}}(s) = E[\max(\hat{Q}(s, a_1), \hat{Q}(s, a_2))]$$
$$\geq \max[E(\hat{Q}(s, a_1), \hat{Q}(s, a_2))]$$
$$= \max[0, 0]$$
$$= V^{\pi}$$

Jensen inequality

[1]Example from Mannor, Simester, Sun and Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. Management Science 2007

Maximization Bias[2] 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:
  $\hat{V}^{\hat{\pi}}(s) = \mathbb{E}[\max \hat{Q}(s, a_1), \hat{Q}(s, a_2)]$
  $\geq \max[\mathbb{E}[\hat{Q}(s, a_1)], [\hat{Q}(s, a_2)]]$
  $= max[0, 0] = V^{\pi}$,
  where the inequality comes from Jensen's inequality.

[2] 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

Initialize $Q_1(s,a)$ and $Q_2(s,a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
loop
  Select a_t using ϵ-greedy $\pi(s) = \arg\max_a Q_1(s_t,a) + Q_2(s_t,a)$
  Observe (r_t, s_{t+1})
  if (with 0.5 probability) **then**
    $Q_1(s_t,a_t) \leftarrow Q_1(s_t,a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, \arg\max_a Q_1(s_{t+1},a)) - Q_1(s_t,a_t))$
  else
    $Q_2(s_t,a_t) \leftarrow Q_2(s_t,a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, \arg\max_a Q_2(s_{t+1},a)) - Q_2(s_t,a_t))$
  end if
  $t = t + 1$
end loop

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

Double Q-Learning

Initialize $Q_1(s,a)$ and $Q_2(s,a)$,$\forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$

loop

  Select a_t using ϵ-greedy $\pi(s) = \arg\max_a Q_1(s_t, a) + Q_2(s_t, a)$

  Observe (r_t, s_{t+1})

  if (with 0.5 probability) **then**

    $Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, \arg\max_a Q_1(s_{t+1}, a)) - Q_1(s_t, a_t))$

  else

    $Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, \arg\max_a Q_2(s_{t+1}, a)) - Q_2(s_t, a_t))$

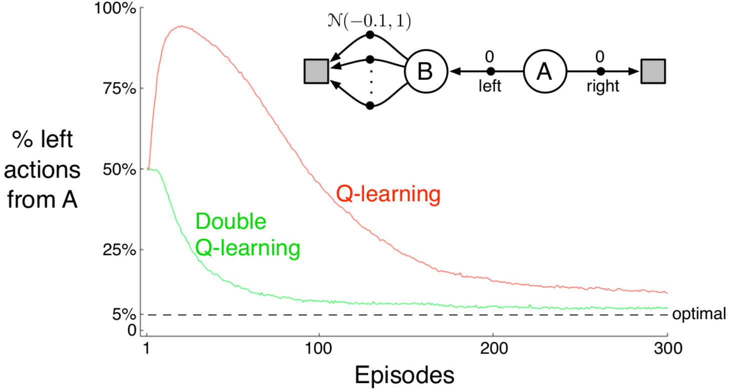
  end if

  $t = t + 1$

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 π 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**