Problem Session Week 4

Markov Decision Processes (MDPs)

Trevor Maxfield
maxfit@stanford.edu

Minae Kwon
minae@cs.stanford.edu

April 28th, 2023

CS 221 - Spring 2023, Stanford University

Reviewing Lecture Material

Reviewing Lecture Material

Defining an MDP

Why Markov Decision Processes?

We don't know... literally!

Why Markov Decision Processes?

We don't know... literally!

- In search $(s, a) \to s'$ every time. Is that realistic?
- Consider walking home from the project session:
    - 90% chance I go directly back to my office.
    - 5% chance I see someone I know and stay and chat.
    - 2% chance I go get a coffee.
    - 1% chance my mom calls me and I'm on the phone for 10 minutes.
    - 1% chance I stop to pet a dog.
    - $\vdots$
    - 0.000...000001% chance the universe ceases to exist.
- How can we put this into our models, keeping the state/action foundation we already have?

Definition of an MDP

Definition
Markov Decision Process

- States: States S and starting state $s_{\text{start}} \in S$.
- Termination State: `isEnd`(s)
- Actions: $a \in A(s)$, possible actions at s.
- Rewards: Reward(s, a, s'), reward from (s, a, s') transition.
- Transitions: $T(s, a, s')$, probability of s' if take a at s.
- (e)Discount: $0 \leq \gamma \leq 1$: discount factor (default 1).

Test yourself: Is Search a special case of an MDP? (Yes, how?)
(Solution: See MDP Lecture 1 Slides 28-30)

Transition Probability Intuition

Which of the following is always true?

1. $\sum_{s \in S} T(s, a, s') = 1$
2. $\sum_{a \in A(s)} T(s, a, s') = 1$
3. $\sum_{s' \in S} T(s, a, s') = 1$

Transition Probability Intuition

Which of the following is always true?

1. $\sum_{s \in S} T(s, a, s') \neq 1$
2. $\sum_{a \in A(s)} T(s, a, s') \neq 1$
3. $\sum_{s' \in S} T(s, a, s') = 1$

Policy

Definition
A policy π is a mapping from each state $s \in S$ to action $a \in A(s)$.

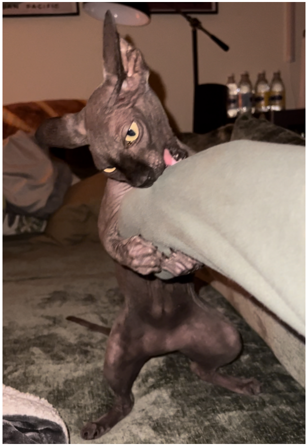
$$\pi : s \in S \rightarrow A(s)$$

Simply tells you what to do at every state. Doesn't have to be deterministic.

Policy

Example policy:

$$\pi_{\text{my cat Norbert}}(s = \text{sees my arm}) = \begin{cases} a_1 = \text{be nice} & \text{with } p = \frac{1}{2} \\ a_2 = \text{bite it} & \text{with } p = \frac{1}{2} \end{cases}$$



Utility vs Value

Following a policy yields a **random path** through the graph $(s_0, a_1, s_1, a_2, s_2, \ldots)$.

$$\text{utility(path)} = \sum_{i=0}^{\infty} \gamma^i \,\text{Reward}(s_i, a_{i+1}, s_{i+1})$$

Definition
The utility of a **policy** is the discounted sum of rewards on the path (making it a random variable).

$$\text{utility}(\pi) = \sum_{i=0}^{\infty} \gamma^i \,\text{Reward}(s_i, \pi(s_i), s_{i+1})$$

How does $0 \leq \gamma \leq 1$ influence utility?

Utility vs Value

Following a policy yields a **random path** through the graph $(s_0, a_1, s_1, a_2, s_2, \ldots)$.

$$\text{utility(path)} = \sum_{i=0}^{\infty} \gamma^i \, \text{Reward}(s_i, a_{i+1}, s_{i+1})$$

Definition
The utility of a **policy** is the discounted sum of rewards on the path (making it a random variable).

$$\text{utility}(\pi) = \sum_{i=0}^{\infty} \gamma^i \, \text{Reward}(s_i, \pi(s_i), s_{i+1})$$

How does $0 \leq \gamma \leq 1$ influence utility?
$\gamma = 1$ emphasizes future, $\gamma = 0$ the current reward.

Utility vs Value

Definition
The value of a policy at a state s_0 is the **expected** utility

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \ \mathrm{Reward}(s_i, \pi(s_i), s_{i+1})\right]$$

Where is the randomness?

Utility vs Value

Definition
The value of a policy at a state s_0 is the **expected** utility

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \text{ Reward}(s_i, \pi(s_i), s_{i+1})\right]$$

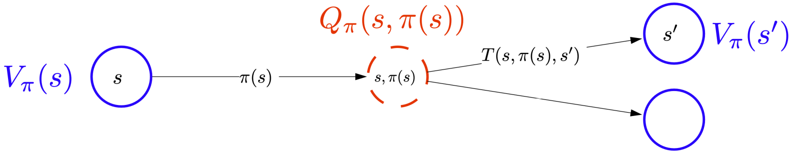
Where is the randomness?

$T(s, a, s')$ and potentially π.

Value and Q-Value

Definition
Let $V_\pi(s)$ be the expected utility received by following policy π from state s.

Definition
Let $Q_\pi(s, a)$ be the expected utility of taking action a from state s, and then following π.



Value and Q-Value

Understanding Q_π(s,a):

Remember that $Q_\pi(s,a)$ is the *expected* utility of taking a from s and then following π. Let's break this up to our current step and the future:

$$\begin{aligned} Q_\pi(s,a) = &\text{Expected reward of taking } (s,a) \\ &+ \text{(discounted) expected future reward} \end{aligned}$$

Expectation (average) is just sum of probability times value! What is the expected future reward? The value of the next state, $V_\pi(s')$.

$$Q_\pi(s,a) = \sum_{s'} T(s,a,s')\text{Reward}(s,a,s') + \sum_{s'} T(s,a,s')\gamma V_\pi(s')$$

$V_\pi(s)$ is just $Q_\pi(s,\pi(s))$ (if s is not an end state).

Solving for Value

By breaking up value of a state into the current step and future steps, we can get:

$$V_\pi(s) = \sum_{s'} T(s, \pi(s), s') \left[\text{Reward}(s, \pi(s), s') + \gamma V_\pi(s')\right]$$

Linear system of equations! (Think of $V_\pi(s)$ as a vector over the states). However, too hard for lots of states, $O(|S|^3)$!

Can we do this approximately in a way that converges to the exact solution?

Reviewing Lecture Material

Algorithms for MDPs

Policy Evaluation

Algorithm: policy evaluation

```
Initialize $V_\pi^{(0)}(s) \leftarrow 0$ for all states $s$.
For iteration $t = 1, \dots, t_{\text{PE}}$:
    For each state $s$:
        $V_\pi^{(t)}(s) \leftarrow \underbrace{\sum_{s'} T(s, \pi(s), s')[\text{Reward}(s, \pi(s), s') + \gamma V_\pi^{(t-1)}(s')]}_{Q^{(t-1)}(s,\pi(s))}$
```

This is just $V_\pi^{(t)}(s) \leftarrow Q_\pi^{(t-1)}(s, \pi(s))$!

Policy Evaluation

Repeat until value stops changing:

$$\max_{s \in S} \left| V_\pi^{(t)}(s) - V_\pi^{(t-1)}(s) \right| \le \epsilon$$

Complexity $O(t_{\text{PE}} S S')$. Why does this work? Fixed point and contraction operator (CS 234).

Optimal Value and Q-Values

All of our discussion so far considered the Value and Q-Value of a specific *policy*. What if we want to optimize our value/policy?

Definition
The optimal value $V_{\text{opt}}(s)$ is the maximum value attained by any policy.

The optimal policy is still a policy!

$$Q_{\text{opt}}(s, a) = \sum_{s'} T(s, a, s') \left[\text{Reward}(s, a, s') + \gamma V_{\text{opt}}(s')\right]$$

Optimal value:

$$V_{\text{opt}}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ \max_{a \in A(s)} Q_{\text{opt}}(s, a) & \text{otherwise} \end{cases}$$

Value Iteration

Remember that $V_\pi(s)$ was just $Q_\pi(s, \pi(s))$ (if not an end state):

$$V_\pi(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ Q_\pi(s, \pi(s)) & \text{otherwise} \end{cases}$$

Policy evaluation found the value of policy by

$$V_\pi^{(t)}(s) \leftarrow Q_\pi^{(t-1)}(s, \pi(s))$$

How can we find the *value* of the optimal policy? Recall:

$$V_{\text{opt}}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ \max_{a \in A(s)} Q_{\text{opt}}(s, a) & \text{otherwise} \end{cases}$$

Value Iteration

Remember that $V_\pi(s)$ was just $Q_\pi(s, \pi(s))$ (if not an end state):

$$V_\pi(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ Q_\pi(s, \pi(s)) & \text{otherwise} \end{cases}$$

Policy evaluation found the value of policy by

$$V_\pi^{(t)}(s) \leftarrow Q_\pi^{(t-1)}(s, \pi(s))$$

How can we find the *value* of the optimal policy? Recall:

$$V_{\text{opt}}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ \max_{a \in A(s)} Q_{\text{opt}}(s, a) & \text{otherwise} \end{cases}$$

$V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in A(s)} Q_{\text{opt}}^{(t-1)}(s, a)$

Value Iteration

Algorithm: value iteration [Bellman, 1957]

```
Initialize $V_{\text{opt}}^{(0)}(s) \leftarrow 0$ for all states $s$.
For iteration $t = 1, \dots, t_{\text{VI}}$:
    For each state $s$:
        $V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in \text{Actions}(s)} \underbrace{\sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}^{(t-1)}(s')]}_{Q_{\text{opt}}^{(t-1)}(s,a)}$
```

Cost: $O(t_{\text{VI}} S A S')$

Optimal Value and Q-Values

Value iteration gives us the *value* of the optimal policy (and by extension Q_{opt}). But what is the actual optimal policy?

At each state, what action maximizes Q_{opt}:

$$\pi_{\text{opt}}(s) = \text{argmax}_{a \in A(s)} Q_{\text{opt}}(s, a)$$

Convergence

When do policy evaluation and value iteration work?

Discount $\gamma < 1$ or MDP graph is acyclic.

Counterexample? $\gamma = 1$ in a 0 reward cycle goes forever.

Reinforce Your Understanding

Policy evaluation: Takes an MDP + π and gives you V_π.

Value iteration: Given an MDP get $(V_{\text{opt}}, Q_{\text{opt}}) \implies \pi_{\text{opt}}$.

Reviewing Lecture Material

Model-Based Methods for MDPs

Unknown Transitions and Rewards: RL

Wouldn't it be nice if life gave you transition probabilities and rewards?

What if we only have:

- Starting state and possible states.
- Actions at each state.
- Termination state.
- Discount factor.

No transition probabilities and rewards. Solution? Reinforcement Learning.

MDS vs Reinforcement Learning

MDPs (offline)

- Have a mental model of the world.
- Find policy to maximize rewards collected.

RL (online)

- Don't know how the world works.
- Perform actions to find out and collect rewards.

Model-Based Value Iteration

As they say, when life gives you no transition or reward function, make your own!

Estimate the missing parts of the MDP (transition and rewards) by exploring the world and use value iteration to find the optimal policy. Hence 'model-based value iteration'.

$$\hat{T}(s,a,s') = \frac{|(s,a,s')|}{|(s,a)|} \quad \hat{R}(s,a,s') = r \text{ from } (s,a,r,s')$$

$$\hat{Q}_{\text{opt}}(s,a) = \sum_{s'} \hat{T}(s,a,s') \left[\hat{R}(s,a,s') + \gamma \hat{V}_{\text{opt}}(s')\right]$$

Model-Based Monte Carlo

How do we explore the space?

Can't miss certain parts of the model (that deterministic π might). Solution? Make sure we randomly explore the space, visiting states infinitely often (in the limit).

Hence 'Model-Based Monte Carlo', randomly traverse the space. (Ergodic theorem gives convergence of $\hat{T}$ and $\hat{R}$ even though the data is not independent)

Why not just skip straight to Q?

Aside: On-Policy vs Off-Policy

Definition

On-Policy: estimate the value of a data-generating (exploration) policy.

Definition

Off-Policy: estimate the value of another policy.

Model-Based Monte Carlo? Off-Policy. We explore the space arbitrarily to find the optimal policy.

Reviewing Lecture Material

Model-Free Methods for MDPs

Model-Free Monte Carlo

In *Model-Based Monte Carlo* we estimated $\hat{T}$ and $\hat{R}$ and used value iteration to compute $\hat{Q}_{\text{opt}}$. What if we just directly estimate $\hat{Q}_{\pi}(s, a)$?

Model-Free Monte Carlo

$\hat{Q}_{\pi}(s_{t-1}, a_t)$ should be the average of u_t, with:

$$u_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

So at $(s_{t-1)}, a_t)$ use the rest of the data to get u_t.

Model-Free Monte Carlo

Equivalent formulation (convex combination)

On each (s, a, u):

$$\eta = \frac{1}{1 + (\#\text{ updates to } (s, a))}$$

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta u$$

Equivalent formulation (stochastic gradient)

On each (s, a, u):

$$\hat{Q}_\pi(s, a) \leftarrow \hat{Q}_\pi(s, a) - \eta[\underbrace{\hat{Q}_\pi(s, a)}_{\text{prediction}} - \underbrace{u}_{\text{target}}]$$

Implied objective: least squares regression

$$(\hat{Q}_\pi(s, a) - u)^2$$

SARSA

Notice that reaching state-action pair (s_{t-1}, a_t) required $u_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$, which is the sum until termination, just for a single update. What if we updated every time we were at (s, a)?

SARSA For each tuple (s, a, r, s', a') in the sequence of our exploration (via π):

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta[r + \gamma \hat{Q}_\pi(s', a')]$$

Interpolate between observed data r and prediction. Bootstrapping! (using the estimate of $\hat{Q}_\pi$ rather than just raw data). Biased but less variance.

Q-Learning

Model-Free Monte Carlo and SARSA only give Q_π. What about Q_{opt}?

We have one really good equation! Bellman Optimality Equation:

$$Q_{\text{opt}}(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V_{\text{opt}}(s')\right]$$

Explore using some policy but learn the optimal policy (off-policy).

Algorithm: Q-learning [Watkins/Dayan, 1992]

On each (s,a,r,s'):

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{\text{opt}}(s,a)}_{\text{prediction}} + \eta\underbrace{(r+\gamma\hat{V}_{\text{opt}}(s'))}_{\text{target}}$$

Recall: $\hat{V}_{\text{opt}}(s') = \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s',a')$

SARSA vs Q-Learning

Algorithm: Q-learning [Watkins/Dayan, 1992]

On each (s, a, r, s'):

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{\text{opt}}(s,a)}_{\text{prediction}} + \eta\underbrace{(r + \gamma \hat{V}_{\text{opt}}(s'))}_{\text{target}}$$

Recall: $\hat{V}_{\text{opt}}(s') = \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s', a')$

Algorithm: SARSA

On each (s, a, r, s', a'):

$$\hat{Q}_{\pi}(s,a) \leftarrow (1-\eta)\hat{Q}_{\pi}(s,a) + \eta[\underbrace{r}_{\text{data}} + \gamma \underbrace{\hat{Q}_{\pi}(s', a')}_{\text{estimate}}]$$

Exploration/Exploitation Trade-off

In Q-Learning we need some policy to generate data while we estimate another policy (the optimal one). Does any policy work?

- Too greedy (always picking the best action) and we won't explore everywhere.
- Too much exploring and we learn too slowly.

Solution? ϵ-greedy policy:

$$\pi_{\text{act}}(s) = \begin{cases} \text{argmax}_{a \in A(s)} \hat{Q}_{\text{opt}}(s, a) & \text{probability } 1 - \epsilon \\ \text{random action from } A(s) & \text{probability } \epsilon \end{cases}$$

Can decay ϵ over time, guarantees we explore and learn.

Reviewing Lecture Material

Q-Learning and Function Approximators

Q-Learning and SGD

We can frame Q-learning as:

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow \hat{Q}_{\text{opt}}(s,a) - \eta \left[\hat{Q}_{\text{opt}}(s,a) - (r + \gamma \hat{V}_{\text{opt}}(s')) \right]$$

Where $\hat{Q}_{\text{opt}}(s,a)$ is a prediction, and $-(r + \gamma \hat{V}_{\text{opt}}(s')$ is a target. Looks like gradient descent (sorta).

Does $\hat{Q}_{\text{opt}}(s,a)$ have any impact on $\hat{Q}_{\text{opt}}(s',a')$?

Q-Learning and SGD

We can frame Q-learning as:

$$\hat{Q}_{\text{opt}}(s, a) \leftarrow \hat{Q}_{\text{opt}}(s, a) - \eta \left[\hat{Q}_{\text{opt}}(s, a) - (r + \gamma \hat{V}_{\text{opt}}(s')) \right]$$

Where $\hat{Q}_{\text{opt}}(s, a)$ is a prediction, and $-(r + \gamma \hat{V}_{\text{opt}}(s')$ is a target. Looks like gradient descent (sorta).

Does $\hat{Q}_{\text{opt}}(s, a)$ have any impact on $\hat{Q}_{\text{opt}}(s', a')$? No!

This is just *independently* memorizing values for (s, a) pairs. Are the pairs truly independent?

Function Approximation

Do not forget: $\hat{Q}_{\text{opt}}(s, a)$ just maps (s, a) to a value estimate.

Instead of having a large table of $\hat{Q}_{\text{opt}}(s, a)$ (as we do currently), we define a function:

$$\hat{Q}_{\text{opt}}(s, a; \mathbf{w}) = \mathbf{w} \cdot \phi(s, a)$$

Algorithm: Q-learning with function approximation

On each (s, a, r, s'):

$$\mathbf{w} \leftarrow \mathbf{w} - \eta[\underbrace{\hat{Q}_{\text{opt}}(s, a; \mathbf{w})}_{\text{prediction}} - \underbrace{(r + \gamma \hat{V}_{\text{opt}}(s'))}_{\text{target}}]\phi(s, a)$$

Reviewing Lecture Material

Summary

Reinforce Your Understanding

Outputs Q_{opt} (Off-Policy):

- **Value Iteration**: $V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in A(s)} Q_{\text{opt}}^{(t-1)}(s, a)$.
- **Model-Based Value Iteration**: Estimate T and R using Monte Carlo, run value iteration using estimates $\hat{T}$ and $\hat{R}$.
- **Q-Learning**: Estimate $\hat{Q}_{\text{opt}}(s, a)$ as an average of reward to s' and estimated optimal max value of s'.

Outputs Q_π (On-Policy):

- **Policy Iteration**: $V_\pi^{(t)} \leftarrow Q_\pi^{(t-1)}(s, \pi(s))$.
- **Model-Free Monte Carlo**: Estimate $\hat{Q}_\pi$ by u.
- **SARSA**: Estimate $\hat{Q}_\pi(s, a)$ by update (s, a, r, s', a') and previous $Q_\pi(s, a)$.

Reinforce Your Understanding

Algorithm	**Estimating**	**Based On**
Model-Based Monte Carlo	$\hat{T}, \hat{R} \implies \hat{Q}_{\text{opt}}$	$s_0, a_1, r_1, s_1, \ldots$
Model-Free Monte Carlo	$\hat{Q}_{\pi}$	u
SARSA	$\hat{Q}_{\pi}$	$r + \hat{Q}_{\pi}$
Q-Learning	$\hat{Q}_{\text{opt}}$	$r + \hat{Q}_{\text{opt}}$

Problem 0: Choosing an Algorithm

Problem 0: Choosing an Algorithm

Choosing an Algorithm

On each of the following slides we are going to choose an algorithm to solve a given problem. Choose from:

- Model-Based Monte Carlo.
- Q-Learning
- Model-Free Monte Carlo.
- SARSA

There may be more than one right answer.

Expensive Experiments

You work in a chemistry lab that is conducting some *extremely* expensive experiments. Unfortunately, sometimes the actions you take cause non-deterministic outcomes (due to unobservable factors), and your reaction transitions to a different state randomly. Your team has several different strategies they'd like to test for running these experiments, but you don't have enough budget for lots of trials. Fortunately, the number of states and possible actions are relatively small, and you have detailed notes on many past experiments run. What should you do and why?

Expensive Experiments

You work in a chemistry lab that is conducting some *extremely* expensive experiments. Unfortunately, sometimes the <u>actions</u> you take cause non-deterministic outcomes (due to unobservable factors), and your reaction transitions to a different <u>state</u> randomly. Your team has several different <u>strategies</u> they'd like to test for running these experiments, but you don't have enough budget for lots of trials. Fortunately, the number of states and possible actions are relatively small, and you have detailed notes on many past experiments run. What should you do and why?

Model-Based Monte Carlo would let us simulate many strategies (policies) using old and new data by generating $\hat{T}$ and $\hat{R}$ before using Policy Iteration.

Poker-Playing Roommates

Your roommate Valentin seems to win a lot of money playing poker against his friends. Based on what you know about him, you've got some ideas for strategies against him, but you're concerned you'll lose to his friends by tailoring your strategy to him. You decide to build a poker bot to test your strategies against random players online. You choose your reward to be how much you win at the end of a round. Which algorithm should you use and why?

Poker-Playing Roommates

Your roommate Valentin seems to win a lot of money playing poker against his friends. Based on what you know about him, you've got some ideas for strategies against him, but you're concerned you'll lose to his friends by tailoring your strategy to him. You decide to build a poker bot to test your strategies against random players online. You choose your reward to be how much you win at the end of a round. Which algorithm should you use and why?

Model-Free Monte Carlo is a good choice since we only get a reward at the terminal state, meaning that SARSA updates would be slower. Since we are testing specific policies we need an on-policy algorithm.

Monte Catlo?

You decide to foster a cat, Monte, from the local Humane Society. Unfortunately, Monte is quite skittish and really likes the dark. As such, he won't get out from under your bed. You've assembled an arsenal of treats, toys, and trinkets to try and lure out Monte. Some things seem to pique his interest, but he won't seem to come out. You're pretty sure that presenting him with the right order of items at the right time of day might convince him to come out. Which algorithm could you use?

Monte Catlo?

You decide to foster a cat, Monte, from the local Humane Society. Unfortunately, Monte is quite skittish and really likes the dark. As such, he won't get out from under your bed. You've assembled an arsenal of treats, toys, and trinkets to try and lure out Monte. Some things seem to pique his interest, but he won't seem to come out. You're pretty sure that presenting him with the right order of items at the right time of day might convince him to come out. Which algorithm could you use?

Q-Learning would work well here, since we'd like to learn an optimal policy (get Monte out from under the bed), and have no idea how he reacts to things. Alternatively, *Monte* Carlo (:

Monopoly

For your CS 221 project you decide to build a bot to play Monopoly. After coding up all of the states, (pieces on the board, how much money each player has, etc.) you decide to train with Model-Free Monte Carlo. After using all of your compute credits, you realize that nearly every entry in your table that stores $\hat{Q}(s, a)$ is empty (how you made a table that big is besides the point). How can you fix this?

Monopoly

For your CS 221 project you decide to build a bot to play Monopoly. After coding up all of the states, (pieces on the board, how much money each player has, etc.) you decide to train with Model-Free Monte Carlo. After using all of your compute credits, you realize that nearly every entry in your table that stores $\hat{Q}(s, a)$ is empty (how you made a table that big is besides the point). How can you fix this?

Q-Learning with function approximation would help here. Two states that differ by only a small amount (say how much money you have, \$100 vs \$110) probably have similar $\hat{Q}$ values. Using a function approximation would allow for generalization to unseen state action pairs.

Problem 1: Riding the Bus

Problem 1: Riding the Bus

Identifying an MDP

Identifying an MDP

Norbert wants to go from their house (located at 1) to the gym (located at n). At each location s, they can either (i) deterministically walk forward to the next location $s+1$ (takes 1 unit of time) or (ii) wait for the bus. The bus comes with probability ϵ, in which case, they will reach the gym in $1+\alpha(n-s)$ units of time, where α is some parameter. If the bus doesn't come, well, they stay put, and that takes 1 unit of time.

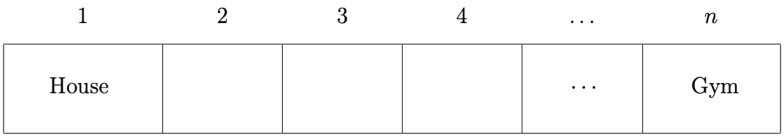


Figure 1: Norbert's Path

Identifying an MDP

How can we model this as an MDP?

- S
- T
- A
- R
- T

Identifying an MDP

How can we model this as an MDP?

- States
- Termination State
- Actions
- Rewards
- Transitions

Identifying an MDP

How can we model this as an MDP?

- States:

Identifying an MDP

How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Norbert's location.
- Termination State:

Identifying an MDP

How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Norbert's location.
- Termination State: $\mathbf{1}[s = n]$
- Actions:

Identifying an MDP

How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Norbert's location.
- Termination State: $\mathbf{1}[s = n]$
- Actions: {Walk, Bus}
- Rewards:

Identifying an MDP

How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Norbert's location.
- Termination State: $\mathbf{1}[s = n]$
- Actions: $\{\text{Walk, Bus}\}$
- Rewards:

$$\text{Reward}(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s + 1 \\ -\infty & \text{otherwise} \end{cases}$$

$$\text{Reward}(s, \text{Bus}, s') = \begin{cases} -1 - \alpha(n - s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

- Transitions:

Identifying an MDP

How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Norbert's location.
- Termination State: $\mathbf{1}[s = n]$
- Actions: {Walk, Bus}
- Rewards: $\text{Reward}(s, a, s')$
- Transitions:

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s + 1 \\ 0 & \text{otherwise} \end{cases}$$

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1 - \epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases}$$

Problem 1: Riding the Bus

Finding Policy Value

Finding Policy Value

Compute a closed form expression for the value of the "always walk" policy and the "always wait for the bus" policy (using some or all of the variables ϵ, α, n). Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s+1 \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s+1 \\ -\infty & \text{otherwise} \end{cases}$$

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1-\epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1-\alpha(n-s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

$V_{\text{Walk}}(s) =$ ______________________________

$V_{\text{Bus}}(s) =$ ______________________________

Finding Policy Value

Compute a closed form expression for the value of the "always walk" policy and the "always wait for the bus" policy (using some or all of the variables ϵ, α, n). Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s+1 \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s+1 \\ -\infty & \text{otherwise} \end{cases}$$

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1-\epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1-\alpha(n-s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

$$V_{\text{Walk}}(s) = -(n-s)$$

$$V_{\text{Bus}}(s) = \underline{\hspace{8cm}}$$

Finding Policy Value

Compute a closed form expression for the value of the "always walk" policy and the "always wait for the bus" policy (using some or all of the variables ϵ, α, n). Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s+1 \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s+1 \\ -\infty & \text{otherwise} \end{cases}$$

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1-\epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1-\alpha(n-s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

$$V_{\text{Walk}}(s) = -(n-s)$$

$$V_{\text{Bus}}(s) = \epsilon(-1-\alpha(n-s)) + (1-\epsilon)(-1+V_{\text{Bus}}(s))$$

Finding Policy Value

Compute a closed form expression for the value of the "always walk" policy and the "always wait for the bus" policy (using some or all of the variables ϵ, α, n). Assume a discount rate of $\gamma = 1$.

$$T(s,\text{Walk},s') = \begin{cases} 1 & \text{if } s' = s+1 \\ 0 & \text{otherwise} \end{cases} \qquad R(s,\text{Walk},s') = \begin{cases} -1 & \text{if } s' = s+1 \\ -\infty & \text{otherwise} \end{cases}$$

$$T(s,\text{Bus},s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1-\epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s,\text{Bus},s') = \begin{cases} -1-\alpha(n-s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

$$V_{\text{Walk}}(s) = -(n-s)$$

$$V_{\text{Bus}}(s) = -\alpha(n-s) - \epsilon^{-1}$$

Finding Policy Value

For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

$$V_{\text{Walk}}(s) = -(n - s) \qquad V_{\text{Bus}}(s) = -\alpha(n - s) - \frac{1}{\epsilon}$$

For walking to be preferable we need:

Finding Policy Value

For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

$$V_{\text{Walk}}(s) = -(n-s) \qquad V_{\text{Bus}}(s) = -\alpha(n-s) - \frac{1}{\epsilon}$$

For walking to be preferable we need: $V_{\text{Walk}}(s) \geq V_{\text{Bus}}(s)$:

Finding Policy Value

For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

$$V_{\text{Walk}}(s) = -(n-s) \qquad V_{\text{Bus}}(s) = -\alpha(n-s) - \frac{1}{\epsilon}$$

For walking to be preferable we need: $V_{\text{Walk}}(s) \geq V_{\text{Bus}}(s)$:

$$n - s \leq \alpha(n-s) + \frac{1}{\epsilon}$$

$$(1-\alpha)(n-s) \leq \frac{1}{\epsilon}$$

Which leads to:

$$\begin{cases} \epsilon \leq \frac{1}{(1-\alpha)(n-s)} & \alpha < 1 \\ \epsilon > 0 & \alpha \geq 1 \end{cases}$$

Problem 1: Riding the Bus

Q Learning

Q Learning

Due to bureaucracy, Norbert's town is unable to provide them with transition probabilities or a reward function (i.e. a bus schedule). To solve this, Norbert decides to use reinforcement learning, specifically Q-learning to determine the best policy. Norbert starts going around town both by bus and by walking, recording the following data:

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Q Learning

Run the Q-learning algorithm once over the given data to compute an estimate of the optimal Q-value $Q_{\text{opt}}(s, a)$. Process the episodes from left to right, assume all Q-values are initialized to zero, and use a learning rate of $\eta = 0.5$ and a discount of $\gamma = 1$.

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Q Learning

Recall the Q-learning update:

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\hat{Q}_{\text{opt}}(s,a) + \eta(r + \gamma \max_{a' \in \text{Action}(s')} \hat{Q}_{\text{opt}}(s',a'))$$

With $\eta = 0.5$ and $\gamma = 1$.

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Find $\hat{Q}(s,a)$ for $s = 1,2,3,4$ and $a \in \{\text{Bus}, \text{Walk}\}$.

Q Learning

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow \frac{1}{2}\hat{Q}_{\text{opt}}(s,a) + \frac{1}{2}(r + \max_{a' \in \text{Action}(s')} \hat{Q}_{\text{opt}}(s',a'))$$

Using the updates:

$(1, \text{Bus}, -1, 1)$:

$(1, \text{Bus}, -1, 1)$:

$(1, \text{Bus}, 3, 3)$:

$(3, \text{Walk}, 1, 4)$:

$(4, \text{Walk}, 1, 5)$:

Q Learning

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow \frac{1}{2}\hat{Q}_{\text{opt}}(s,a) + \frac{1}{2}(r + \max_{a' \in \text{Action}(s')} \hat{Q}_{\text{opt}}(s',a'))$$

Using the updates:

$$(1, \text{Bus}, -1, 1) : \hat{Q}(1, \text{Bus}) = 0.5(0) + 0.5(-1 + 1 \max(0,0)) = -0.5$$
$$(1, \text{Bus}, -1, 1) : \hat{Q}(1, \text{Bus}) = 0.5(-0.5) + 0.5(-1 + 1(\max(0,-0.5))) = -0.75$$
$$(1, \text{Bus}, 3, 3) : \hat{Q}(1, \text{Bus}) = 0.5(-0.75) + 0.5(3 + 1(\max(0,0))) = 1.125$$
$$(3, \text{Walk}, 1, 4) : \hat{Q}(3, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0,0))) = 0.5$$
$$(4, \text{Walk}, 1, 5) : \hat{Q}(4, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0,0))) = 0.5$$

All other $\hat{Q}(s,a) = 0$.

Problem 1: Riding the Bus

Other Policies and Algorithms

Choose a Different Algorithm

After using Q-learning, Norbert wants to try a different algorithm to compute $\hat{Q}$. They consider SARSA, Model-Free Monte Carlo, and Model-Based Monte Carlo with Value Iteration. Which of these can Norbert use to compute the *optimal* policy? Explain why.

Choose a Different Algorithm

After using Q-learning, Norbert wants to try a different algorithm to compute $\hat{Q}$. They consider SARSA, Model-Free Monte Carlo, and Model-Based Monte Carlo with Value Iteration. Which of these can Norbert use to compute the *optimal* policy? Explain why.

Model-Based Monte Carlo. SARSA and Mode-Free Monte Carlo are on policy, can only give you the value of a specific policy, not the optimal one.

Free Bus Ride

Norbert's job decided to start subsidizing their bus pass. Norbert decides that for states $s < \frac{3}{4}n$ they will take the bus and for states $s \geq \frac{3}{4}n$ they will walk. Write down a policy function $\pi(s)$ to represent this.

Free Bus Ride

Norbert's job decided to start subsidizing their bus pass. Norbert decides that for states $s < \frac{3}{4}n$ they will take the bus and for states $s \geq \frac{3}{4}n$ they will walk. Write down a policy function $\pi(s)$ to represent this.

The policy is

$$\pi(s) = \begin{cases} \text{Bus} & \text{if } s < \frac{3}{4}n \\ \text{Walk} & \text{if } s \geq \frac{3}{4}n \end{cases}$$

Free Bus Ride

Norbert wants to know how long their expected commute will be. Should they use Q-Learning, SARSA, or Model-Free Monte Carlo to compute that, and why? Will the algorithm converge to the correct $\hat{Q}_\pi(s, a)$ *for all states* after going to work enough times? (recall that the starting state is $s = 1$ and assume $n > 1$).

Free Bus Ride

Norbert wants to know how long their expected commute will be. Should they use Q-Learning, SARSA, or Model-Free Monte Carlo to compute that, and why? Will the algorithm converge to the correct $\hat{Q}_\pi(s, a)$ *for all states* after going to work enough times? (recall that the starting state is $s = 1$ and assume $n > 1$).

Q-Learning will not work in this case since it computes the optimal policy. The other two will only update $\hat{Q}_\pi(1, \text{Bus})$ since starting state is $s = 1$ and $\pi(1) = \text{Bus}$.

Convergence Issues

Why is there (potentially) a convergence problem in this situation? How can Norbert change their policy to π' so that $\hat{V}_{\pi'}(s)$ converges for *all* states?

Convergence Issues

Why is there (potentially) a convergence problem in this situation? How can Norbert change their policy to π' so that $\hat{V}_{\pi'}(s)$ converges for *all* states?

The current policy will only generate data at state $s = 1$. Their π needs to visit all of the states, can be non-deterministic.

Getting to the Gym on Time

Now that Norbert understands on vs off policy learning, they consider their attempt at Q-learning from part (b). They'd like to generate more data, but still get to the gym in a reasonable amount of time. What kind of policy would do this while guaranteeing convergence?

Getting to the Gym on Time

Now that Norbert understands on vs off policy learning, they consider their attempt at Q-learning from part (b). They'd like to generate more data, but still get to the gym in a reasonable amount of time. What kind of policy would do this while guaranteeing convergence?

ϵ-greedy policy would balance the two goals and guarantee convergence (we'd visit every state infinitely often!)

Bonus: Another MDP

Bonus: Another MDP

Speedbumps

See problem 3 in the handout.