Introduction to Machine Learning
(Fall 2022)

Lab attendance check

Type in your section passcode to get attendance credit (within the first fifteen minutes of your scheduled section).
Passcode: Enter

Group information

Create/join a group

1. One person (and only one) should create a group.
2. Everyone else should enter the group name below (in the form `groupname_0000`).

Join group: Submit

To join another group or leave your current group, reload the page.

You are not in a group.

Reinforcement Learning

Due: Monday, December 05, 2022 at 11:00 PM

For this lab:

- You may find it useful to refer to the notes on Reinforcement Learning.
- Don't spend too long on any one question - use the help queue if you are stuck!

1) Q-Learning Warmup

In this week's lab, we explore the Q-learning algorithm with different methods of setting up rewards.

1.1) Q-Learning vs. Value-Iteration

Before proceeding, it is important to note the differences between the value iteration (VI) algorithm in the MDP notes versus the Q-learning (QL) algorithm in the Reinforcement Learning notes to be explored in this week's lab.

1.1.1)

What is the *principal difference* between VI and QL algorithms?

Pick one:

- ○ **i)** VI computes and outputs a value function; QL learns and outputs a policy.
- ○ **ii)** VI computes and outputs a policy; QL learns and outputs a value function.
- ◉ **iii)** While running, VI has access to the true state transition function T and reward function R of the MDP, while QL does not.
- ○ **iv)** While running, QL has access to the true state transition function T but not the reward function R of the MDP, while VI has access to both.

Submit View Answer **100.00%**

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.1.2)

In which of these settings would you use QL, because VI is not applicable? Assume that the sizes of the state space for all of these are small enough so that tabular VI and QL are in principle applicable.

Choose the Q-learning applications:

- ☐ **i)** Finding shortest paths to a goal for a robot in a grid with perfect transitions (perfect in the sense the exact probability is completely known).
- ☐ **ii)** Finding paths to a goal with shortest expected length for a robot in a grid with noisy transitions with known transition probabilities, given the state at each time step.
- ☑ **iii)** Finding paths to a goal with shortest expected length for a robot in a grid with unknown noisy transition probabilities, given the state at each time step.
- ☐ **iv)** Playing a single-person video game, where we know the transition probabilities of the game at each time step given the state of the game and a score
- ☑ **v)** Playing a single-person video game, whose transition probabilities we don't know, given at each time step the state of the game and a score.

Submit View Answer **100.00%**

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.2) Q-Learning algorithm with different reward functions

We explore the Q-learning algorithm using a deterministic MDP with two possible actions (b and c) and four states ($s0$, $s1$, $s2$, $s3$). The transition model is

$$\begin{aligned} T(s0, b, s1) &= 1 \\ T(s0, c, s2) &= 1 \\ T(s1, *, s0) &= 1 \\ T(s2, *, s3) &= 1 \\ T(s3, *, s3) &= 1 \end{aligned}$$

where $*$ represents either b or c.

Some notes:

1. All other transitions have probability 0.
2. The goal state is $s2$, and $s3$ is a terminal state (similar to $ in the grid-world question we have looked at).
3. Assume that if there are ties in the Q function for actions b and c in a state, then we pick action b.
4. Assume the Q function is initialized to 0 for every state-action pair and that, after every episode (sequence of actions) of length 10, the agent is restarted in state $s0$.
5. Assume a learning rate (α) of 0.5.

6. Assume an ϵ-greedy strategy with $\epsilon = 0$.
7. Assume a discount factor of 1.

Note that we restart the agent in state $s0$ after every 10 steps because otherwise it may reach $s3$ and stay there forever.

1.2.1)

Draw a state transition diagram, like you drew in the MDP lab.

Next, we will see two typical schemes for setting up rewards in domains with a goal state: Goal-reward based, and Stochastic-shortest-path (SSP) based, and we will study how each scheme affects the exploration of the state space.

Goal-reward

In the goal-reward case, every action taken from the goal state $s2$ gives an immediate positive reward of 1, and leads to a zero-reward next state (in fact terminal state) $s3$ that can never be escaped. Taking any action from any state other than the goal state provides zero reward. In other words, we have a reward function which outputs 0 for every state-action pair, except for $R(s2, *) = 1$.

1.2.2)

What action will be selected the first time the agent encounters $s0$? And why (explain why during the checkoff)?

- (●) b
- ( ) c

Submit View Answer **100.00%**

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.2.3)

What action will be selected the second time the agent encounters $s0$? And why (explain why during the checkoff)?

- (●) b
- ( ) c

Submit View Answer **100.00%**

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.2.4)

What action will be selected the hundredth time the agent encounters $s0$? And why (explain why during the checkoff)?

- (●) b
- ( ) c

Submit View Answer **100.00%**

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

Stochastic shortest path

In the stochastic shortest path (SSP) case, we put zero reward on taking any action from the goal state. Every action taken from the goal state $s2$ leads to a zero-reward terminal state $s3$ that can never be escaped. We put -1 in rewards everywhere else. In other words, we have a reward

function which outputs -1 for every state-action pair except $R(s2, *) = R(s3, *) = 0$.

1.2.5)

What action will be selected the first time the agent encounters $s0$? And why (explain why during the checkoff)?

- ○ b
- ○ c

Submit | View Answer | Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.2.6)

What action will be selected the second time the agent encounters $s0$? And why (explain why during the checkoff)?

- ○ b
- ○ c

Submit | View Answer | Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.2.7)

What action will be selected the hundredth time the agent encounters $s0$? And why (explain why during the checkoff)?

- ○ b
- ○ c

Submit | View Answer | Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

2) Q-Learning in a 2D grid

Now, we'll take a look at Q-learning in a simple 2D grid setting but with a single goal location. We'll adopt the same state space and action space as in MDP lab.

Specifically, our state space is a 10-by-10 grid, with the bottom-left state indexed as state $(0, 0)$, and our four possible actions are moving North, South, East or West (note that if the robot tries to move off the board, it cannot; it just stays in the same state). Our single goal is at state $(3, 6)$.

Remember that for Q-learning, the transition probabilities and reward values are not known in advance by the method—the agent just gets to observe state, action, and reward values as it moves through the domain.

Some notes (please read!):

- A new episode is started by sampling the first state uniformly at random.
- The agent follows an epsilon-greedy policy with $\epsilon = 0.1$.
- Every action taken from the goal state leads to a zero-reward state that can never be escaped. Thus, to continue learning, we repeat the steps above. Note that we start a new/reset episode only after the agent reaches the goal state.
- In the case of a tie in the value $\max_a Q(s, a)$ across actions a, we choose the "best" action randomly.
- All transitions are noisy; that is, there is some non-zero probability of the agent moving somewhere different from its desired destination. For example, say the agent in in state (0,0) and takes a "North" action, there is a non-zero chance that it actually ends up in state (1,1).
- Our γ (discount factor) is set to 0.9 and our α (learning rate) is set to 0.5.

Note that the scale of the colors changes across the different plots, per the bar on the right of each plot.

2.1) Goal reward

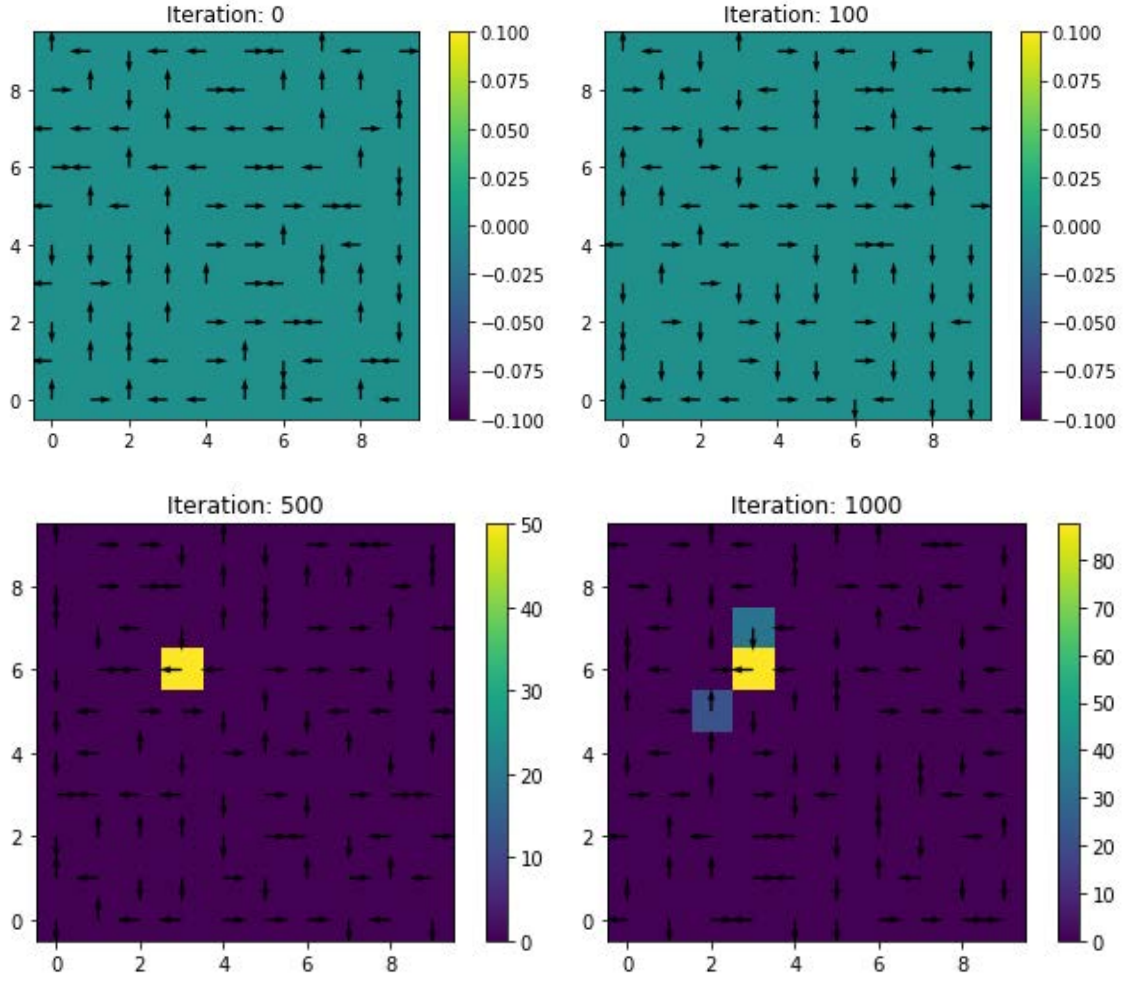
2.1.1)

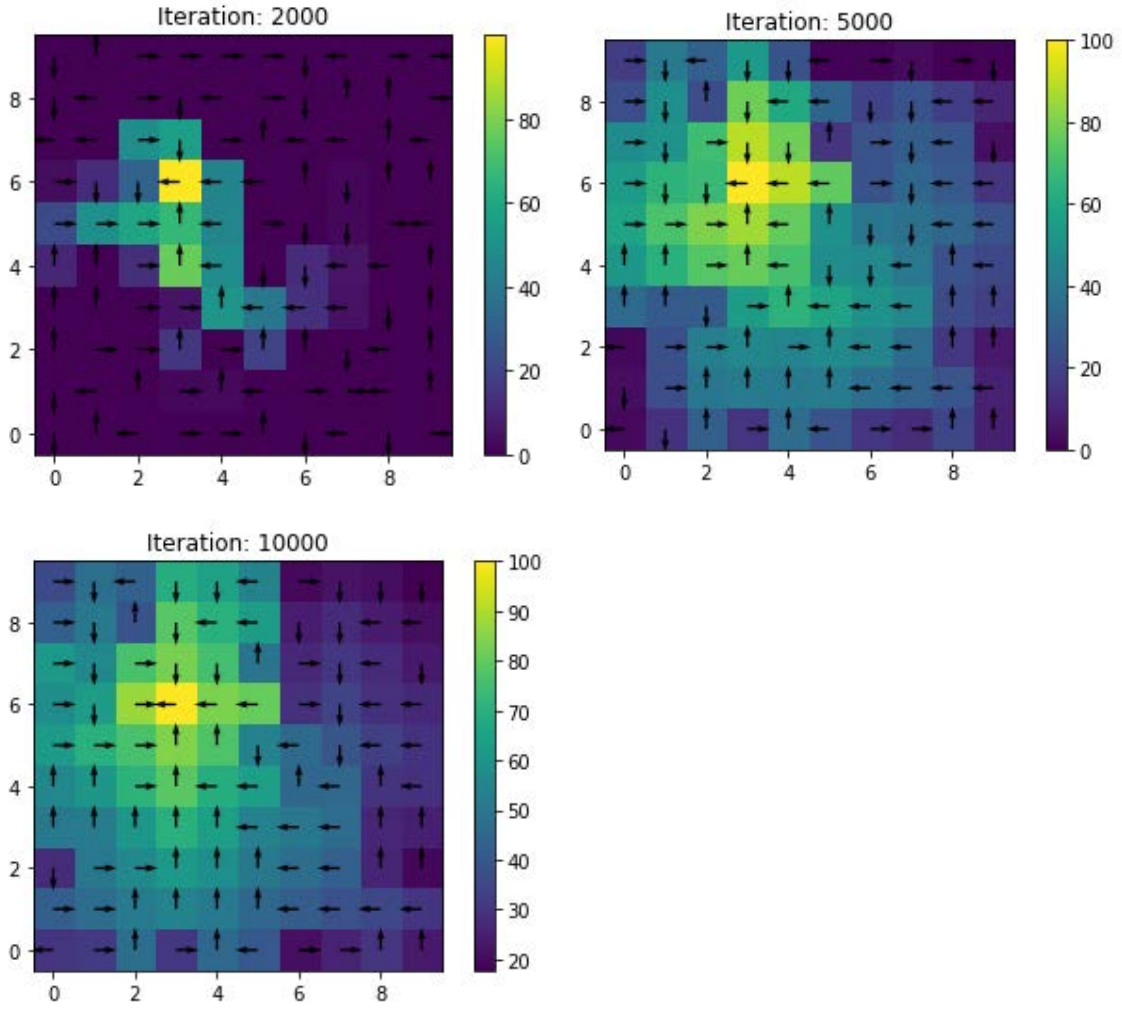
Recall that we are interested in learning the value of taking the best actions starting at a given state. In Q learning, we try to learn these values. But for the moment in this sub-question, suppose we actually knew these "true" $\max_a Q(s, a)$ values exactly. Also suppose that taking an action in the goal state yields a reward of 100. What is the highest value (a number) for the function $\max_a Q(s, a)$ among all state-action input pairs? Roughly what is the $\max_a Q(s, a)$ value when s is a direct neighbor of the goal state (to answer this, ignore the error probabilities and imagine that the actions always take the robot to the intended location)?

2.1.2)

Now let's go back to Q-learning (where we don't know the true values of states and want to learn them).

Below are plots of the maximum Q values $\max_a Q(s, a)$ at different points during an execution of Q learning, with the iteration number shown in the title.





What has been happening for the first 100 steps of Q-learning?

2.1.3)

At iteration 500, why does the state at (3, 6) have value 50?

2.1.4)

At iteration 1000, how many more times do you think the agent reached the goal state?

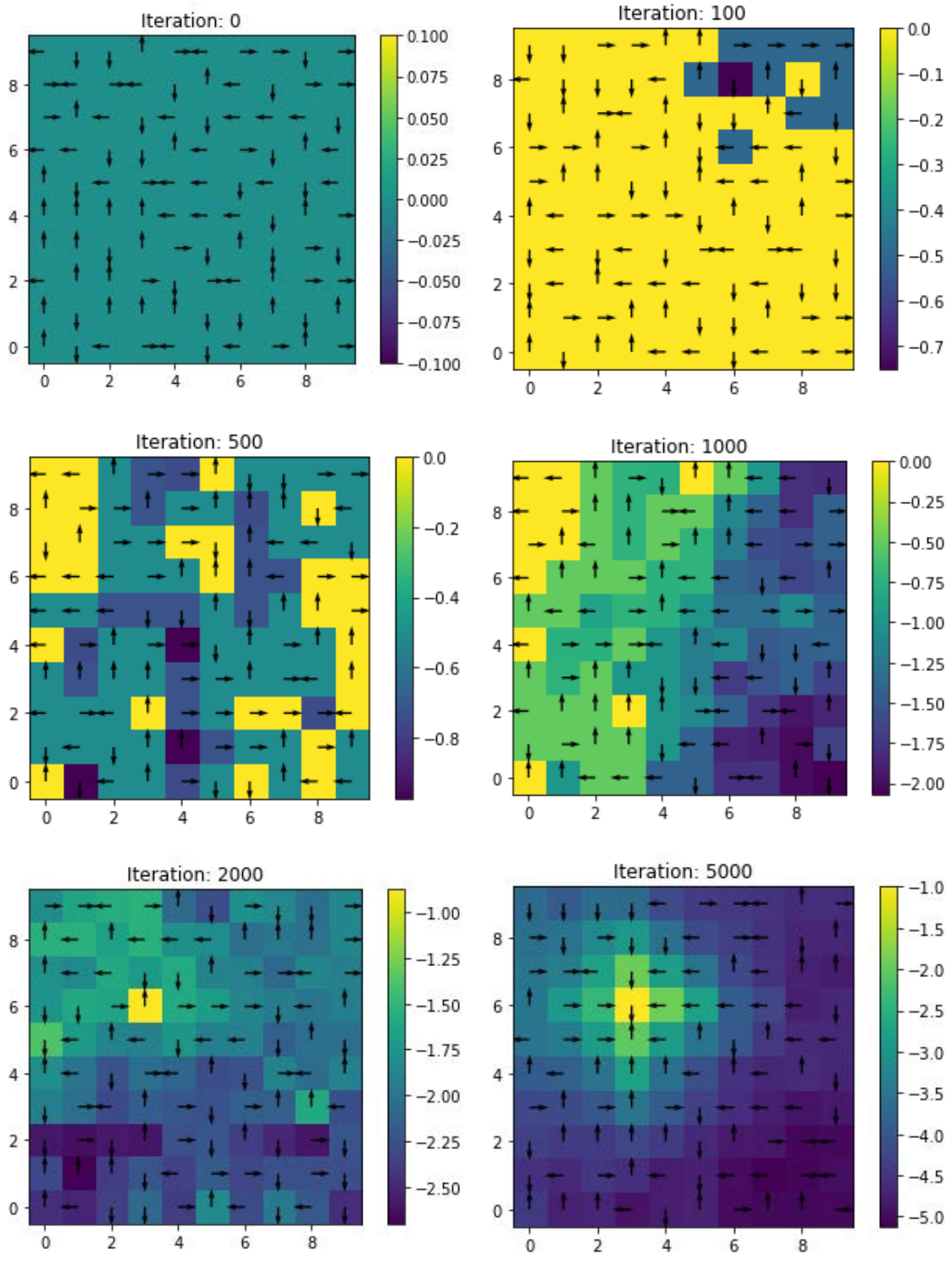
2.1.5)

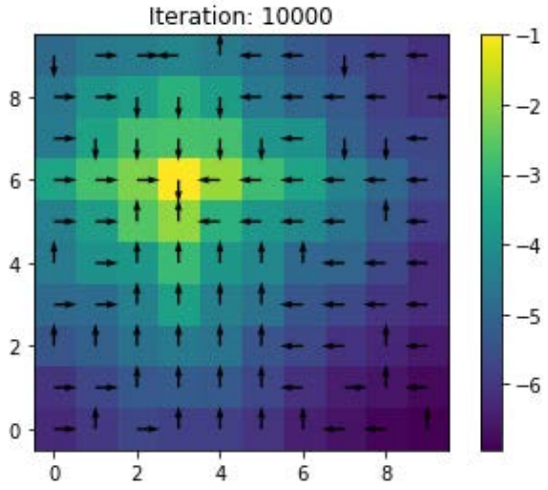
At iteration 10,000, how close are the values plotted (our estimates of the true value of taking the best actions starting at the given state) to the actual value of taking the best actions starting from the given state?

In particular, what should the value of the bottom-right corner be? (To make this easier to think about, you can assume that the transitions are deterministic; that is, the robot always moves in the direction it is "aiming".)

2.2) Stochastic shortest path

These are plots of the maximum Q values of the states $\max_a Q(s, a)$ using the SSP formulation as we run 10,000 iterations of Q-value learning, plotting at specific iterations. **Note that the scale of the colors changes across the different plots, per the bar on the right of each plot.**


Iteration: 0
Iteration: 100
Iteration: 500
Iteration: 1000
Iteration: 2000
Iteration: 5000



2.2.1)

At iteration 100, this is a pretty strange picture. At first we thought we had bugs! Remember that the squares are colored according to the *maximum* of $Q(s, a)$ over all actions in that state, and that they are all initialized to 0. What can we say about the squares that are colored yellow? What about the squares colored blue?

2.2.2)

At iteration 100, how can it be possible for (6, 6) to be colored blue, when all its neighbors are yellow?

2.2.3)

At iteration 100, is there a state where we have tried all the actions twice?

2.2.4)

At iteration 1000, there is a yellow block in the upper left and lots of the policy arrows are pushing the agent to go up there. Why? Is that desirable behavior in this problem? What if there is more than one goal state?

2.2.5)

At iteration 10,000, how close are the values plotted (our estimates of the true value of taking the best actions starting at the given state) to the actual value of taking the best actions starting from the given state? Roughly what should the value in the bottom right corner be? (Assume that the transitions are deterministic.)

Checkoff 1:
Have a check-off conversation with a staff member, to explain your answers.

Ask for Help | Ask for Checkoff

3) No Exit

Recall that the hyperparameter epsilon (ϵ) characterizes a trade-off between exploration and exploitation in reinforcement learning. When we use an "ϵ-greedy" strategy in Q learning, we take a completely random action with probability ϵ; and with probability $1 - \epsilon$, we take the action that'd lead to the highest Q value, i.e. we take $\arg\max_a Q(s, a)$.

We'll explore how choosing the value of epsilon affects the performance of Q learning in a very simple game.

3.1) Intuition

The choice of epsilon can affect the overall behavior of Q-learning. Let's consider three possible values for epsilon: 0.0, 0.5, and 1.0.

3.1.1)

Which of these epsilon values risks **never** finding the optimal policy? --

Submit | View Answer | Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

3.1.2)

Which of these epsilon values has the highest risk of spending way too much time exploring parts of the space that are unlikely to be useful? --

Submit | View Answer | Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

3.1.3)

Which of these epsilon values is guaranteed to cause optimal behavior **during** learning? --

Submit | View Answer | Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

3.2) Experience

For this part, you will use a Colab notebook we have prepared for you. You can find the Colab notebook here.

Once you run the code, wait patiently until you see a yellow and purple square on a teal background (you may need to scroll down from the "score" and "reward" text lines printed out). Ignore everything else for now. Click play in the button right below the square. This is a movie of a policy playing the game No Exit. It's kind of like Pong: the purple square is the "ball" and the yellow square is your "paddle". The actions are to move the paddle up, down, or keep it still.

The state is specified by the positions and velocities of the ball and paddle, with a special added "game over" state.

The transition model is a very approximate physics model of the ball reflecting off walls and the paddle, except if the ball gets past the paddle in the positive x direction, the game is over.

The agent gets a reward of +1 on every step it manages to survive.

When watching the game play out, you'll sometimes see that the purple square gets near the right-hand border and then suddenly it changes to a state with the purple square in the bottom left and the yellow one in the upper right -- this means that the game terminated and then reset to the initial state.

Now we can go back and look at the other output in the notebook:

- First, we print what happens during learning in the format (number of iterations, average score): after every 10 iterations of batch Q learning, we take the current greedy policy and run it to see what its average score is. This score represents how long the episode ran before the ball ran off the map, or 100 if it lasted for that long.
- Next is a plot of the score as a function of number of iterations.
- Finally, we run the greedy policy with respect to the last Q-value function for 10 games and report the rewards achieved on each game. We also show a movie clip from a handful of these 10 games.

3.2.1)

What is the maximum possible score in the game? What is the minimum possible score?

3.2.2)

Run the code given on the notebook for values of ϵ in the set $0, 0.5, 1$. Does your observation match your answers from 3.1?

Remember that this is a small instance, so sometimes the random noise of the environment might prevent you from seeing any useful information. Run the notebook two or three times if something doesn't line up with your expectation, and then ask for help.

4) Value Alignment

Last week we first learned about the problem of "value alignment", which aims to ensure the values embedded in AI agents are aligned with human values.

4.1)

If a reward function is value-aligned (to a specific individual's values), is an agent which used this reward function to learn necessarily also aligned? Why or why not?

4.2)

Philosophers argue over how AI agents should be aligned to human values. Assume, again, that we only want to create an agent which is aligned to a specific person's values. Should this agent be aligned to:

- The human's instructions?
- The human's intentions?
- The human's expressed behaviors?
- Behaviors which are in the 'best interest' for the human?

And why?

To make this concrete, you might want to think about recommendations - like those of Netflix. Think about a person who aspires to watch more "good" films. This person might search for Oscar-winning films (their instructions). But perhaps they're really looking for Cannes-winning films (their intent), but they often spend their time watching reality TV instead (their expressed behaviors). Perhaps Netflix thinks it is actually in their best interest to watch more documentaries.

(An article from Iason Gabriel at DeepMind has some nice discussions and formalization of this topic: Goal Misgeneralization)

4.3)

In our consideration of the value alignment problem thus far, we have made a big assumption: that we only want to create agents which align with a specific individual's values. In practice, agents will inevitably interact with many people with different value systems, and from different cultures and religions with different values.

How might you try to design a reward function or agent in the face of conflict across different value systems? What are the pros and cons of your proposal?

Consider, for example, writing a reward function for an autonomous vehicle. In a terrible scenario, an autonomous vehicle might have to choose between saving one of two people: one an elder and one a child. Research has shown that people from Western cultures often express a preference for saving the child, while Eastern cultures express a preference for saving the elder. How might we think about reconciling these preferences? This is an open question, so we're not looking for 'correct' answers. Instead, we're looking for thoughtful, creative answers!

Checkoff 2:
Have a check-off conversation with a staff member, to explain your answers.

Ask for Help | Ask for Checkoff

Further discussions of these questions:

This study of moral preferences for autonomous vehicles is based off of research from MIT!

- Moral Machines
- How East and West Differ on Who a Self Driving Car Should Save

Further reading:

- Artificial Intelligence, Values, and Alignment
- AI Alignment, Philosophical Pluralism, and the Relevance of Non-Western Philosophy - LessWrong (A talk transcript by MIT PhD student Tan Zhi Xuan)

Food for (lighter) thought

This blog post includes a little game that illustrates **multi-armed bandits**, which are non-sequential (environment resets after each action) reinforcement-learning problems. You can compare how well you do against some classic approaches to solving these problems. And, you can help save the world from Bieber.

RES.TLL-008 Social and Ethical Responsibilities of Computing

Spring 2023