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.

For this lab:

- You will need to understand the material in the course notes on Markov decision processes.
- Don't spend too long on any one question - use the help queue if you are stuck!

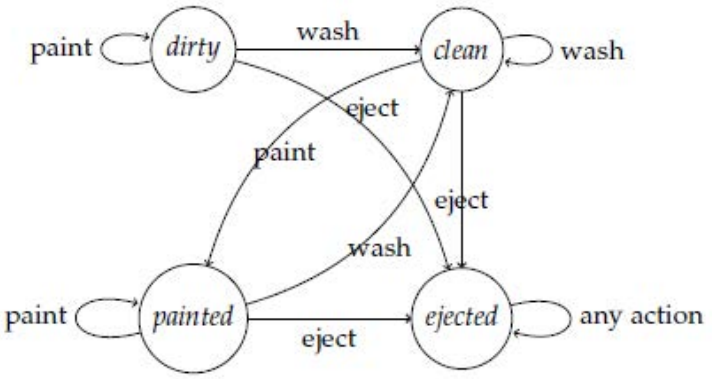
1) Deterministic Wash & Paint MDP

We will model aspects of a very simple wash and paint machine as a Markov decision process (MDP). An agent controls the actions taken, while the environment responds with the transition to the next state.

Our simple machine has three possible operations: "wash", "paint", and "eject" (each with a corresponding button). Objects are put into the machine. Each time you push a button, something is done to the object. The machine has a camera inside that can clearly detect what is going on with the object and will output the state of the object: dirty, clean, painted, or ejected.

In this question, you will devise a policy that will take the state of an object as input and select a button to press until finally you press the eject button. You get reward +10 for the "eject" action on a painted object, reward 0 for "eject" action on an object that is dirty or clean, reward 0 for any action on an ejected object, and reward -3 otherwise.

We start out with a brand-new machine that operates reliably and deterministically, as illustrated in the state diagram below. Here state transition arcs are labeled with the action responsible for the transition. Specifically, when we "wash" a dirty or painted object, it becomes clean; and when we "paint" a clean object it becomes painted. If we "eject" a dirty, clean, or painted object, it becomes ejected. An ejected object stays ejected, for any action.



1.1)

You will formulate the brand-new machine as an MDP, but with a deterministic transition function. It will be useful to write out and save tables/diagrams of your answers and show them to staff during the checkoff.

Write out a specification of the **state space** and **action space**.

The **transition model** is specified by the diagram above, where an arc from state s to state s' under action a indicates a transition probability $T(s, a, s') = 1$, and lack of an arc from s to s' indicates a transition probability $T(s, a, s') = 0$.

The **reward function**, $R(s, a)$, needs your definition. Given m states and n actions, the reward matrix should be m by n, and will look something like:

```
          wash   paint  eject
dirty      --     --     --
clean      --     --     --
painted    --     --     --
ejected    --     --     --
```

For the matrix in the response below, order the states as "dirty", "clean", "painted", and "ejected": and the actions as "wash", "paint", "eject".

Enter the reward matrix as a list of lists:

Check Formatting | 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)

Suppose we have an infinite horizon and discount factor 1. That means you can take as many steps as you want to, and all rewards are weighted equally whether they happen at the beginning or end of the action sequence. What would be the optimal action to take in each state? What would your total sum of rewards be (under the optimal policy) if you started in state dirty?

1.3)

Suppose we have a horizon of 2 and discount factor 1. That means you could only take two steps in total. Would the policy from the previous question change? Why, or why not? What would your total sum of rewards be (under the optimal policy) if you started in state dirty?

1.4)

Suppose we have an infinite horizon and discount factor 0.5. Let's also assume that under our action policy, we always "paint" an object if it is clean, and always "eject" an object if it is painted. What would be the sum of discounted future rewards if you start with a dirty object in each of the policies listed next:

- Your policy is to "eject" an object whenever it is dirty?

- Your policy is to "wash" an object whenever it is dirty, followed by optimal actions after that?

Which is the better policy?

2) Stochastic Wash & Paint MDP

Our wash and paint machine has gotten old, and is no longer so reliable. Now, many of its state transitions are stochastic in response to specific actions:

Wash:

- If you perform the "wash" operation on any object, whether it's dirty, clean, or painted, it will end up clean with probability 0.9 and dirty otherwise.

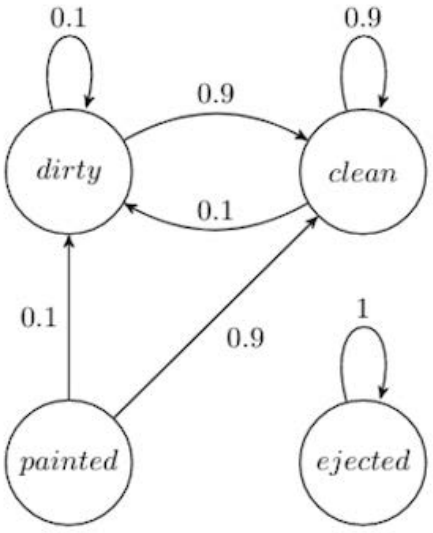
Paint:

- If you perform the "paint" operation on a clean object, it will become nicely painted with probability 0.8. With probability 0.1, the paint misses but the object stays clean, and also with probability 0.1, the machine dumps rusty dust all over the object and it becomes dirty.
- If you perform the "paint" operation on a painted object, it stays painted with probability 1.0.
- If you perform the "paint" operation on a dirty object, it stays dirty with probability 1.0.

Eject:

- If you perform an "eject" operation on any object, the object comes out of the machine and this fun game is over. The object remains ejected regardless of any further action.

Here is an example state diagram for the "wash" operation, now with arcs labeling the probability $T(s, a, s')$, for action a being "wash".



2.1)

In order to visualize the stochastic machine MDP, draw the state diagrams for the "paint" and "eject" operations.

2.2)

The state space, action space, and reward function remain the same as for our deterministic machine, but now our **transition model** has changed. Write out the transition matrices for the "wash", "paint", and "eject" actions. Specifically, provide three transition matrices (rows should sum to one, with the conventions used in this class), one transition matrix for each of the "wash", "paint", and "eject" actions.

Given m states, each transition matrix should be m by m, corresponding to $T(s, a, s')$ with row indicating s and column s'. Thus, a transition matrix for a given action a will look something like:

```
          dirty  clean  painted  ejected
dirty      --     --      --       --
clean      --     --      --       --
painted    --     --      --       --
ejected    --     --      --       --
```

For the prompt just below, order the rows and columns as dirty, clean, painted, ejected.

Enter the transition matrix as a list of lists for "wash" action:

Check Formatting | 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.

2.3)

We next consider some finite horizon scenarios (and with discount factor $\gamma = 1$), but now with our stochastic wash & paint MDP.

2.3.1)

First, assume horizon $h = 1$. What is our optimal action a^1 if you are in a painted state? What is the optimal action-value function Q^h for this case, $Q^1(s = \text{painted}, a^1)$?

2.3.2)

If we instead have a finite horizon of $h = 2$, what is our optimal action a^2 if we are in a painted state with two steps to go? Why is this different than (or the same as) a^1? What is $Q^2(s = \text{painted}, a^2)$; how does this compare to $Q^1(s = \text{painted}, a^1)$ and why?

2.3.3)

Still with a finite horizon of $h = 2$, now we start in state clean. In the deterministic (brand new) wash & paint machine, we saw that the optimal action was to paint, then eject, for a total reward of 7.

Now, with horizon $h = 2$ in our stochastic wash & paint machine, we still hope to paint then eject. Will this $Q^2(s = \text{clean}, a^2 = \text{paint})$ have a reward of 7, more than 7, or less than 7? Why?

2.4)

We switch to an infinite-horizon scenario. For our stochastic machine, here is the infinite-horizon Q function (computed via value iteration) for γ near 1.

```
              wash        paint     eject
dirty    [[ 2.32274541 -0.70048204  0.        ]
clean     [ 2.32274541  5.71581775  0.        ]
painted   [ 2.32274541  6.9        10.        ]
ejected   [ 0.          0.          0.        ]]
```

What is the optimal thing to do with a clean object?

What will you do if it becomes dirty?

Does this optimal policy make intuitive sense?

2.5)

If the machine became much less reliable (i.e., washing and painting only achieve the desired transition, say, 20% of the time), how do you think the optimal policy would change?

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

Ask for Help | Ask for Checkoff

3) Grid-World -- Finite Horizon Q Values

In the previous problem, we only had four states: dirty, clean, painted, and ejected. In this problem, we use a two-dimensional "grid-world" with a robot in it; we have a state for every square on the grid, representing the robot's location.

Our four possible actions are moving North, South, East or West (note that the robot cannot move off the board, so some states have fewer possible actions). The transitions are also substantially noisy; when commanding a move to a target state, there is a 50% probability of moving to the target state, and a $(50/g)\%$ probability of landing instead at **one of the g vertical or horizontal neighbor states of the target state**. In most cases, g will be four, but at the boundaries of our grid-world, g and the available transitions will include only the neighboring squares that are within grid-world.

We want to identify the best action that the robot can take in each state (i.e., the best policy), if it had a specific horizon h of remaining moves. We do not have any discount for future moves: the discount factor $\gamma = 1$.

Consider a grid world with its "floor plan" represented as a list of strings of characters:

```
['.........',
 '........*.',
 '.........',
 '.........',
 '.........',
 '.........',
 '.........',
 '.........',
 '.$.......',
 '.........']
```

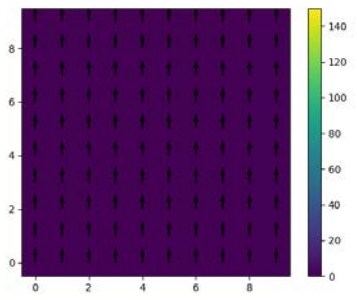
Each character corresponds to a square in the grid. The meanings of the characters are:

- '.' : a normally habitable square, from which the robot can move.
- '$' : a **terminal state**. Reward is 100 for any action from this state, and then the game immediately ends.
- '*' : a **teleporter state**. Reward is 50 for any action from this state, and the **next state is chosen uniformly at random** from all occupiable states, including $ and *. This reward can be claimed multiple times.

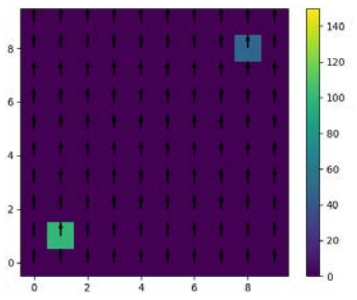
Below are plots of the Q^h values of the states as we consider horizons h from 0 to 99. The color for each square s represents $Q^h(s, a^{opt})$, where a^{opt} is the optimal finite-horizon action to take at horizon h, i.e., $a^{opt} = \arg\max_a Q^h(s, a)$. The arrows represent the optimal move, a^{opt}, pointing N, S, E, or W.

Note: Before proceeding, confirm that you understand what these diagrams mean, and how they relate to Q^h in the finite-horizon set up in the course notes on MDPs. What equations in the notes apply to this scenario? Ask for help if you're not sure!

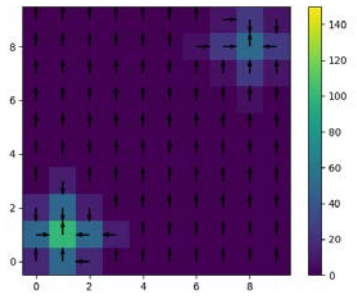
0



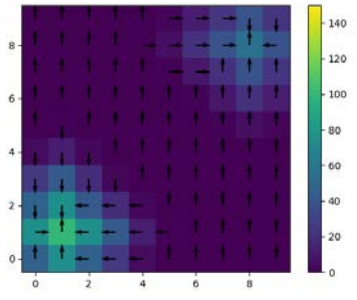
1



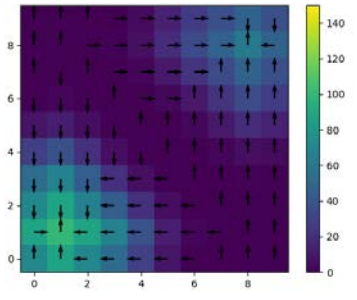
2



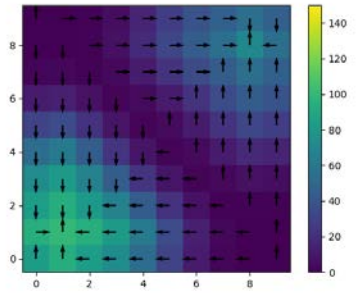
3



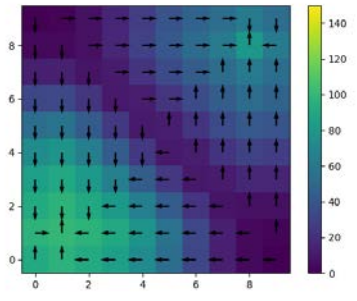
4



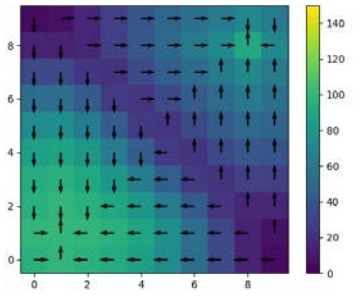
5



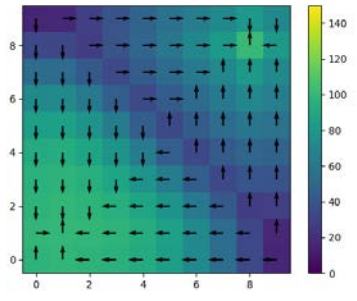
6



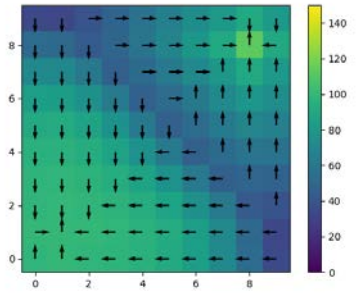
7



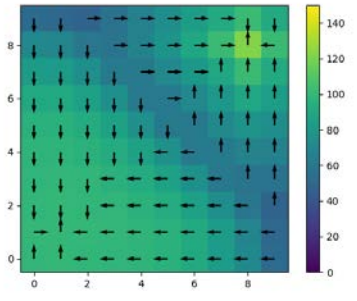
8



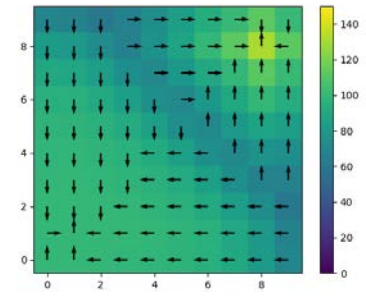
9



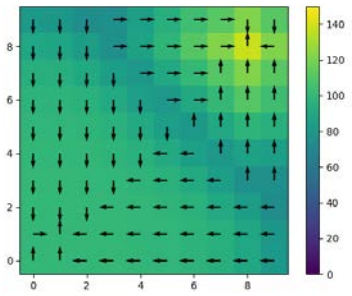
10



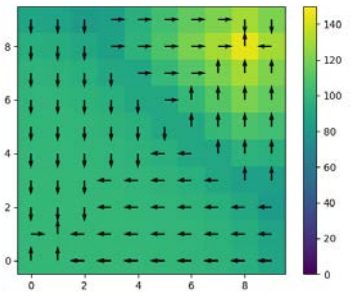
11



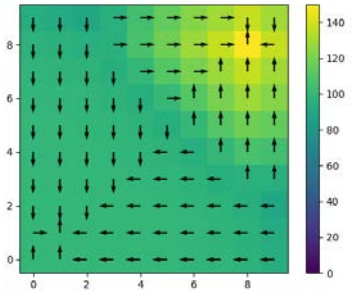
12



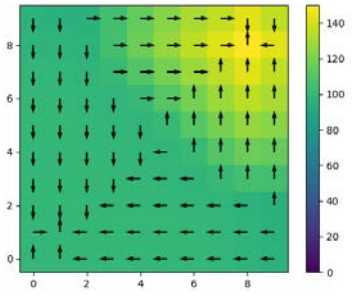
13



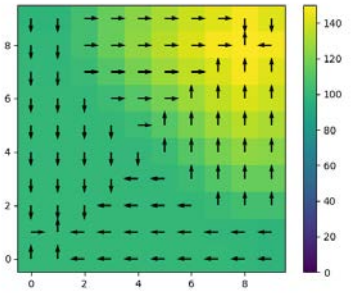
14



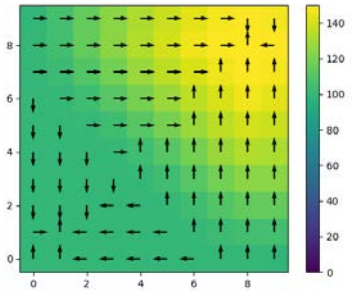
15



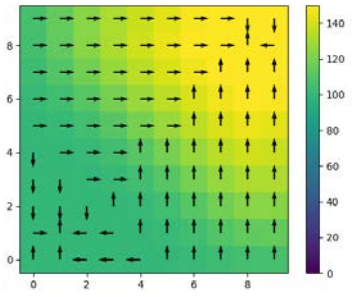
16



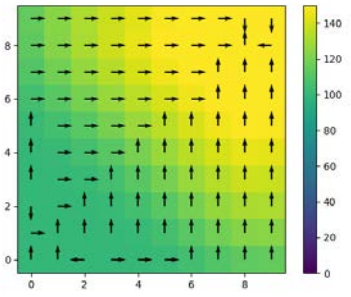
17



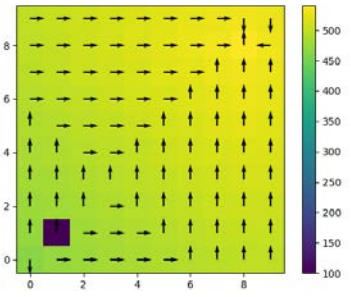
18



19



99 (note scale change!)



3.1)

In the first picture all the values are zero, and with a default "best" action, N. Why?

The second picture considers an horizon $h = 1$ scenario. In this case, we have two states with non-zero Q values. What states are they? What values do they reflect?

3.2)

What is new in the horizon 2 scenario?

How is it possible that there are non-zero Q^2 for states that do not neighbor $ and *?

3.3)

What happens for horizons roughly in the range of 3 to 7? What do you observe about the upper right and lower left portions of the grid?

3.4)

Look at the horizon 11 scenario. Why does the upper right teleporter state * now have higher Q^{11} value than the lower left terminal state, $?

3.5)

What's happening to the arrows in the pictures corresponding to horizon values near $h = 16$?

3.6)

What's happening around horizon 19?

The value of being near the "teleporter" is about 150. Why? Give an informal description of how an optimal policy plays out for states near the teleporter, with the estimated Q^h values at this point.

3.7)

The last picture is for $h = 99$ (note the change in scale for the color bar!). If we were to consider scenarios well beyond horizon 99, what would the Q^h value function look like for large h?

For very long horizons, do we expect the game to eventually terminate?

3.8)

In thinking through each successive horizon above, we built our Q^h value based on knowing our Q^{h-1} value. If we were to run the infinite horizon value iteration algorithm (in pseudocode at the end of the chapter on MDPs) with $\gamma = 1$, and were to plot the estimate of Q and a' after each iteration, what would those plots look like?

4) Reward Hacking

Reward functions and discount factors define a task and the optimal solutions to this task. We introduce the "Value Alignment Problem", which concerns the challenge of aligning the preferences encoded by reward functions (and discount factors) with human preferences.

4.1)

CoastRunners is a boat racing game. This video shows how the game should be played.

OpenAI added this game to their internal testing suite for reinforcement learning algorithms, and they used the game's score as the reward function. They found their learned agents achieved scores ~20% higher than human players; a success. This video shows an example gameplay for their agent.

How is it possible that an agent which scores better than humans repeatedly crashes into targets, and does not make progress toward the goal?

4.2)

Carla is an open source urban driving simulator that aims to support the development, training, and validation of autonomous driving systems. This simulator formulates driving as an MDP, and has the following (simplified) reward function and discount factor:

$$r = (1)\Delta d + (0.05)\Delta v + (-0.00002)\Delta c$$

$$\gamma = 0.99$$

- Δd is the change in distance traveled in meters along the shortest path from start to goal, regularly calculated using the car's current position;
- Δv is the change in speed in km/h;
- Δc is the change in collision damage, expressed in range [0, 1], where 0 is no damage and 1 is damage when the car crashes.

What information do you need to encode your state? Does this reward function align with your preferences, and why?

Hint: think about how you might compare a successful trajectory, a motionless trajectory where the car does not move, and/or an unsuccessful trajectory where the car crashes $(\Delta c = 1)$.

4.3)

Let's suppose we wanted an autonomous vehicle to observe speed limits. Is this a good idea? How might we add something like this into our RL system?

4.4)

Recall that our reward functions in their general form depend only on state s and action a. This means that the reward function does not depend on the history of states. Do you think alignment (say, to a specific individual's preferences) is always possible with a reward that depends only on the current state? Does this limitation affect what aspects of the situation you'd want to include in the problem state?

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

Ask for Help | Ask for Checkoff

Further Reading

You can read more about the (common!) problem of reward mis-design in autonomous driving in this recent paper. (You might also ask: should we even be trying to formulate the autonomous driving task this way in the first place?)

We will continue to explore this topic of value alignment next week. In the meantime, if you're interested in learning more about misspecification of reward functions, these academic papers are a good place to start!

- Concrete Problems in AI Safety
- The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models
- Here is a long un-curated list of "objective-hacking" in a variety of learning contexts.

RES.TLL-008 Social and Ethical Responsibilities of Computing

Spring 2023