

HOMWORK 5: REINFORCEMENT LEARNING AND UNSUPERVISED LEARNING

10-701 Introduction to Machine Learning
(PhD) (Spring 2026)

Carnegie Mellon University

<https://piazza.com/cmu/spring2026/10701>

OUT: March 30th, 2026*

DUE: April 8th, 2026 11:59 PM

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., “Jane explained to me what is asked in Question 2.1”). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section in our course syllabus for more information: <https://www.cs.cmu.edu/~10701-s26/index.html#Syllabus>
- **Late Submission Policy:** See the late homework policy here: <https://www.cs.cmu.edu/~10701-s26/index.html#Syllabus>
- **Submitting your work to Gradescope:** There will be two submission slots for this homework on Gradescope, the Written and the Programming:
 - For the written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using the written submission slot. Please use the provided template. The best way to format your homework is by using the Latex template released in the handout and writing your solutions in Latex. However submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Each derivation/proof should be completed in the boxes provided below the question, **you should not move or change the sizes of these boxes** as Gradescope is expecting your solved homework PDF to match the template on Gradescope. If you find you need more space than the box provides you should consider cutting your solution down to its relevant parts, if you see no way to do this, please add an additional page at the end of the homework and guide us there with a ‘See page xx for the rest of the solution’.

Regrade requests can be made after the homework grades are released, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For \LaTeX users, use ■ and ● for shaded boxes and circles, and don’t change anything else. If an answer box is included for showing work, **you must show your work!**

*Compiled on Tuesday 31st March, 2026 at 02:55

1 K-Means [14 pts]

Recall that in K-means clustering we attempt to find K cluster centers $\mu_j \in \mathbb{R}^D$, $j \in 1, \dots, K$ such that the square of the total distance between each datapoint and the nearest cluster center is minimized. In other words, we attempt to find μ_1, \dots, μ_K that minimizes

$$\sum_{i=1}^N \min_j \|x^{(i)} - \mu_j\|^2$$

where N is the number of data points. To do so, we iterate between assigning $x^{(i)}$ to the nearest cluster center and updating each cluster center μ_j to the average of all points assigned to the j^{th} cluster.

In general, minimizing the above equation for a fixed K is an NP-hard problem. However, it can be solved in polynomial time if the data points are single dimensional ($D = 1$). For this question, we will focus on that case.

1. **[8 pts]** Assume we sort our data points such that $x^{(1)} \leq x^{(2)} \leq \dots \leq x^{(N)}$. Prove that an optimal cluster assignment has the property that each cluster corresponds to some interval of points. That is, for each cluster j there exists i_1, i_2 such that the cluster consists of $x^{(i_1)}, x^{(i_1+1)}, \dots, x^{(i_2)}$.

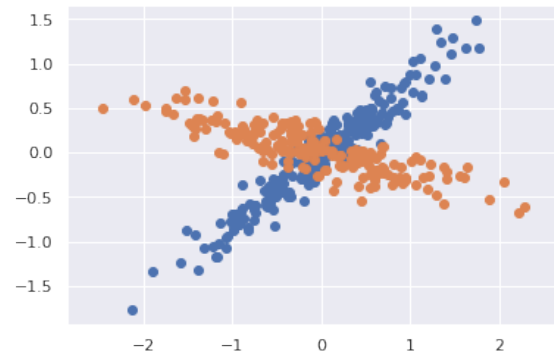
2. **[6 pts]** Develop an $O(KN^2)$ dynamic programming algorithm for single dimensional K-means. [Hint: using the result from the previous part, all that we need to optimize are $K - 1$ cluster boundaries where the i^{th} boundary marks the largest data point in the i^{th} cluster.]

2 PCA [15 pts]

1. [3 pts] Principal component analysis is a dimensionality reduction method that projects a dataset into its most variable components. You are given the following 2D datasets, draw the first and second principle components on each plot. Be sure to label which vector is the first principal component and which is the second.



(a)



For the following questions, consider the following 6 data points in \mathbb{R}^5 , represented as rows in a 6 x 5 matrix X below:

$$X = \begin{bmatrix} -2 & -2 & -2 & 0 & 0 \\ -2 & -2 & -2 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & -2 & -2 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. [1 pts] Is X centered?

☐ Yes ☐ No

3. [3 pts] Write X in its SVD form (a product of three matrices) where the dimensions of the three matrices are as small as possible such that X may be recovered.

Note: SVD decompositions are not unique. To help with grading (and with the rest of PCA below), write the left matrix with columns that have L2 norm equal to one and the right matrix with rows that have L2 norm equal to one. You may calculate the values however you like.

4. [2 pts] What is the first principal component of the data set?

5. [3 pts] If we project the original data set onto 1-D space using the first principal component, what is the variance of the projected data?

Optional space to show your work:

6. [3 pts] For the projected data in the previous part, what is the reconstruction error?

Optional space to show your work:

3 Reinforcement Learning [24 points]

Consider an environment in which our agent requires caffeine to function (if it helps, you can think of the agent as a graduate student). Because caffeine is so important to our agent, we would like the agent to find a policy that will always take the shortest path to coffee.

In order to apply reinforcement learning techniques such as value iteration and policy iteration, we first need to model this scenario as a Markov decision process (MDP). Note that there may be many different ways to define each of the MDP components for a given problem. Recall that an MDP is defined as a tuple (S, A, P, R, γ) where:

S is the (finite) set of all possible states.

A is the (finite) set of all possible actions.

P is the transition function $P : S \times S \times A \rightarrow [0, 1]$, which maps (s', s, a) to $P(s'|s, a)$, i.e., the probability of transitioning to state $s' \in S$ when taking action $a \in A$ in state $s \in S$. Note that $\sum_{s' \in S} P(s'|s, a) = 1$ for all $s \in S, a \in A$.

R is the reward function $R : S \times A \times S \rightarrow \mathbb{R}$, which maps (s, a, s') to the real-value reward $R(s, a, s')$, i.e., the reward obtained when taking action $a \in A$ in state $s \in S$ and arriving at state $s' \in S$.

γ is the discount factor, which controls the relative importance of short-term and long-term rewards. We generally have $\gamma \in [0, 1)$ (i.e., $0 \leq \gamma < 1$), where smaller values mean greater discounting of future rewards.

For this problem, we represent the state of the agent at any time by the triple (x, y, o) , where x and y are the horizontal and vertical coordinates of the agent's location, respectively, and o is the agent's orientation which is one of $\{N, E, W, S\}$ (North, East, West, South). The agent's current orientation is the only direction it can move. The agent is able to move forward, turn right, or turn left (deterministically). All actions are available in all states. More specifically, the action space is $|A| = \{F, R, L\}$ where:

- F (move Forward): The agent moves forward one cell along its current orientation, changing the agent's location but not its orientation.
- R (turn Right): The agent turns right, changing the agent's orientation but not its location.
- L (turn Left): The agent turns left, changing the agent's orientation but not its location.

Walls are represented by thick black lines. The agent cannot move through walls. If the agent attempts to move through a wall, it will remain in the same state.

When the agent reaches the coffee cup in any orientation, the episode ends. Another way to think of this is that every action in the coffee cup state keeps the agent in the coffee cup state.

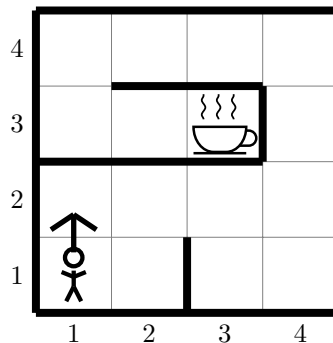


Figure 3.1: MDP for Problem 1, Part I. The goal is for the agent to have a policy that always leads it on the shortest path to the coffee at coordinate $(3, 3)$. In this instance, the agent's current state is $(1, 1, N)$

Consider the instance shown in Figure 3.1. The goal, displayed as a coffee cup, is located at (3, 3). Using the above problem description answer the following questions:

1. [1 point] i.) How many states are in this MDP?

- [1 point] ii.) How many actions are in this MDP?

- [1 point] iii.) What is the total number of entries in the transition table of $T(s' \mid s, a)$?

2. [3 points] Fill in the following probabilities for the transition function P .

		s'			
s	a	(1,2,N)	(1,1,S)	(1,4,N)	(1,4,S)
(1,1,S)	F				
(1,1,N)	F				
(1,4,E)	R				

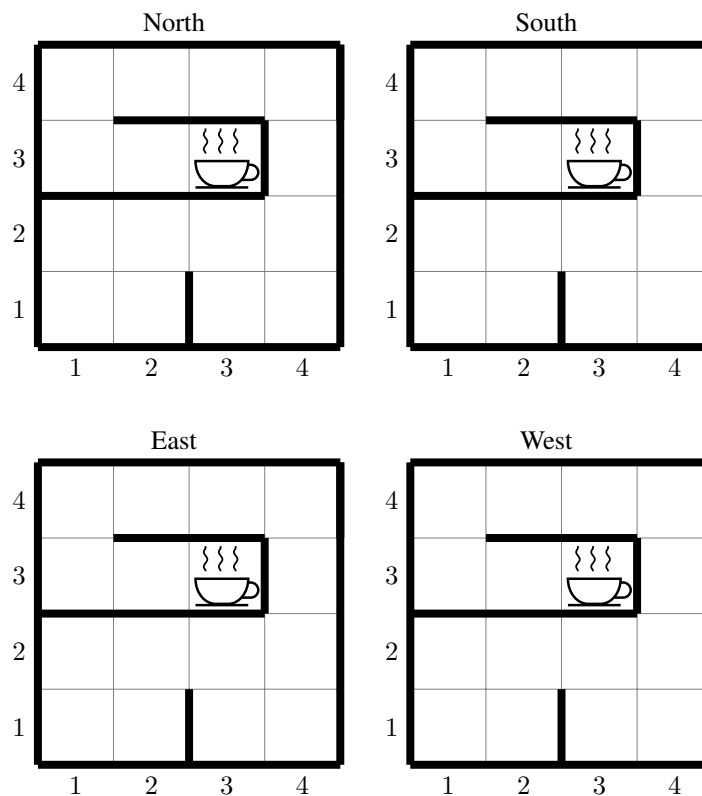
3. [2 points] i.) Propose a suitable reward function $R : S \times A \rightarrow \mathbb{R}$ for this MDP.

[2 points] ii.) Does the value of $\gamma \in (0, 1)$ affect the optimal policy in this setting? Briefly justify your answer.

4. [1 point] How many possible deterministic policies are there, including both optimal and non-optimal policies?

5. [4 points] Show one optimal deterministic policy for this MDP by filling in the 4 grids below (one for each orientation). You should label each cell with one of $\{F, R, L\}$, for Forward, turn Right and turn Left, respectively.

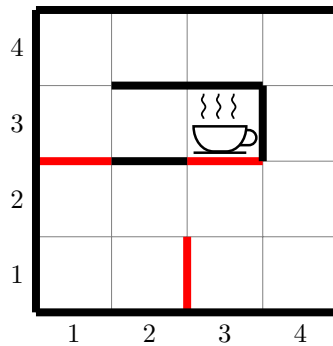
(If there are multiple optimal policies, then give just one. The action chosen for the “coffee” state, $(3, 3, o)$ for $o \in \{N, E, W, S\}$, is arbitrary, so you can just keep the cup symbol there.)



6. Q-Learning

After many unsuccessful attempts at navigating to maze to find caffeine, the agent takes two major decisions.

- Tired of endlessly walking, the agent decides that from now on, it will explore this maze by skateboard. This mode of transportation is significantly faster than walking, so our agent will now always move **two blocks at a time**. For example, taking the action F in the state (1, 2, E) will result in the agent being in state (3, 2, E).
- Noticing the highly unoptimized layout of its environment, the agent decides to break three walls. The new environment is shown below (the red walls have been destroyed and can now be skateboarded over).



To best adapt to this new environment, you decide to use Q-Learning. Assume the learning rate $\alpha = 0.5$, the discount factor $\gamma = 0.9$ and the following rewards:

- The agent gets a reward of 10 for reaching the coffee cup
- The agent gets a reward of -1 every time its movement goes through a broken wall, as the concrete debris damage the wheels of its skateboard

Because skateboarding is hard, *transitions are no longer deterministic*. You *do not know* the dynamics exactly, but instead have to look at sampled trajectories. Initialize the Q function to zero again. Then for each of the episodes (trajectories) below, compute the updated Q values. (Keep in mind that each computation below needs to take previous updates into account, and that all Q -values are initialized to 0.)

[1 point] i.) Beginning at the (1, 1, N) state, the agent skateboards action F and lands in (1, 3, N). It gets -1 reward for going over a broken wall. What is the updated Q value at the starting state?

$$Q((1, 1, N), F) =$$

[1 point] ii.) Continuing from the (1, 3, N) state, the agent takes action R and is now in (1, 3, E). What is the update?

$$Q((1, 3, N), R) =$$

Homework 5: Reinforcement Learning and Unsupervised Learning

[1 point] iii.) Beginning at the (1, 3, E) state, the agent takes action F and is now in (3, 3, E). It gets 10 reward for finding the coffee cup.

$Q((1, 3, E), F) =$

[1 point] iv.) If the agent started a new run at (1, 1, N) and was given the choice between actions F and R, what would the agent choose now? (Note that the agent is not actually choosing this action and is not updating the Q function.) $a =$

[1 point] v.) Now the agent starts a new episode at the (1, 3, N) state. The agent takes action R and is now in (1, 3, E).

$Q((1, 3, N), R) =$

[1 point] vi.) Now the agent starts a new episode at the (1, 1, N) state. The agent skateboards action F and lands in (1, 3, N). It gets -1 reward for going over a broken wall.

$Q((1, 1, N), F) =$

[1 point] vii.) Now if the agent started a new run at (1, 1, N) and was given the choice between actions F and R, what would the agent choose? (Again, do not actually update the Q function.) $a =$

[2 points] viii.) If transitions *were* deterministic, the optimal sequence of actions from (1, 1, N) would be F, R, F. However, we no longer have that guarantee. Assuming all other transitions in all other states are deterministic, give a simple example of transition probabilities for the actions at (1, 1, N) for which the learned optimal move at (1, 1, N) is not F.

4 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment?
(b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

2. (a) Did you give any help whatsoever to anyone in solving this assignment?
(b) If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. (a) Did you find or come across code that implements any part of this assignment?
(b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).