Released: 3rd Feb Deadline: 16th Feb

Problem Set 2

INF266- Reinforcement Learning
Spring 2026

IMPORTANT: Submission of this problem set is **compulsory** to be admitted to the final exam and graduate from the course.

Part I. Problems

In the first part (*Problems*), you are required to provide answers for all the *obligatory* questions, that is, questions marked by a star (*). The remaining questions are optional; however, you are invited to solve them because: (i) they are stepping-stones towards the obligatory questions; (ii) they are similar to the questions you will be asked at the exam.

1. Markov Models and Bellman Equations

Exercise1 Markov Processes

Consider the following scenario: a *Mars Rover* module has been designed to autonomously explore the surface of Mars. Every ten minutes, the module runs its program to transition to a new state. If it is in the normal operative mode of exploration, it will remain in that state with 70% chance and it will keep exploring for the next ten minutes; otherwise it will select uniformly at random among the following three alternatives: taking pictures of the sky, taking samples from the ground, recharging. If it selects taking pictures, it will spend ten minutes taking snapshots of the skies; then, it will revert to exploration with 60% chance or it will take samples from the ground with the remaining probability. If it ends up taking samples from the ground, it will dig for ten minutes; afterward, it will go back to normal operation mode with 35% chance, recharge with 30% chance, dig for more samples with 30% chance, or experience a malfunctioning with 5% chance. If it selects recharging, it will stay put for ten minutes collecting energy through its solar panels; after that, it will revert to normal exploration mode. If it experiences a malfunction, the robot will just shutdown.

1. (*) How can this model comply with the assumption of Markovianity?
2. (*) Express the problem as a Markov Process. Write down its definition and draw its graph.
3. If the robot is in the normal operative mode, are all the states reachable? Is there any state that compromises the reachability of other states?
4. If the robot is in the normal operative mode, what is the probability of malfunctioning in the next 10 minutes? In the next 20 minutes? In the next 30 minutes?
5. (*) How would the model change if you were to be told that the probability of experiencing a malfunction in the digging state would depend on the number of times the robot has been extracting samples?

Consider the *tic-tac-toe*[1] game, where the players are hard-coded computer programs.

6. Can you express this model as an MP? (You do not need to write out explicitly the graph).
7. How many states do you count? (Approximate answers are fine) What does this tell you about scalability?

Exercise2 Markov Reward Processes

Enrich the previous *Mars Rover* scenario with the following rewards: 0 for normal exploration, +5 for taking pictures of the skies, +20 for collecting samples from the ground, 0 for recharging, −100 for malfunctioning.

1. (*) Redraw the graph for the previous MP as a graph for this new MRP.
2. If the robot is in the normal operative mode, what is the reward you would expect it to collect in the next 10 minutes? What is the probability of a negative reward within 20 minutes?

Exercise3 Markov Decision Processes

Reconsider the previous MRP for the *Mars Rover* scenario.

1. (*) Can you model it as a MDP? How would you now model actions, states, transitions and policies?
2. Consider the two objectives of maximizing rewards and minimizing the probability of failure. Assuming the same transitions as in the MRP, would the optimal policy for the two objectives be the same? Would the answer depend on the time horizon?
3. (*) Reconsider the multi-armed bandit problem, for instance, the first problem in Exercise 8 of Problem Assignment 1. Can you formalize it as a MDP? How would it be?
4. In class, we have discussed how MPs and MRPs captures important aspects of RL, but they are not sufficient to model meaningful RL problems. We have then chosen MDPs as the *minimal* models that capture all the relevant aspects of RL that we may care about. Is there any aspect of what you would consider an interesting RL problem that is not captured by MDPs?

[1] `https://en.wikipedia.org/wiki/Tic-tac-toe`

Exercise4 Bellman equations

In class we discussed Bellman equations for MDPs.

1. (*) Can you define Bellman equations for a MP? If yes, draw the corresponding backup diagrams and derive the Bellman equations. If not, explain why not.
2. (*) Can you define Bellman expectation equations for a MRP? If yes, draw the corresponding backup diagrams and derive the Bellman equations. If not, explain why not.
3. (*) Can you define Bellman optimality equations for a MRP? If yes, draw the corresponding backup diagrams and derive the Bellman equations. If not, explain why not.
4. In class we also derived the state-value Bellman expectation equation for an MDP, but we skipped the whole derivation of the action-value Bellman expectation equation. Write out the full derivation from $q_\pi(a, s) = E[G^{(t)}|S^{(t)} = s, A^{(t)} = a]$ to $q_\pi(a, s) = E[R^{(0)} + \gamma q_\pi(a', s')|S^{(t)} = s, A^{(t)} = a]$
5. Consider state-value Bellman expectation equations. They define a linear system. Can you rewrite them in matrix form? (Hint: start from the Bellman expectation equation and isolate a matrix for the rewards and a matrix for the transitions.)
6. Why and when would be useful to express Bellman expectation equations in matrix form?

Exercise5 Optimality

1. Is the value $v_\pi(s)$ of a state s an absolute/objective value? If not, what does it depend upon?
2. Must an optimal policy always be deterministic?
3. What is the implication of a deterministic policy for the exploration-exploitation trade-off?
4. Assume I run an optimal policy π^* and an alternative policy π'. Is the optimal policy always guaranteed to achieve the highest return against π' on every run?

Part II.
Report

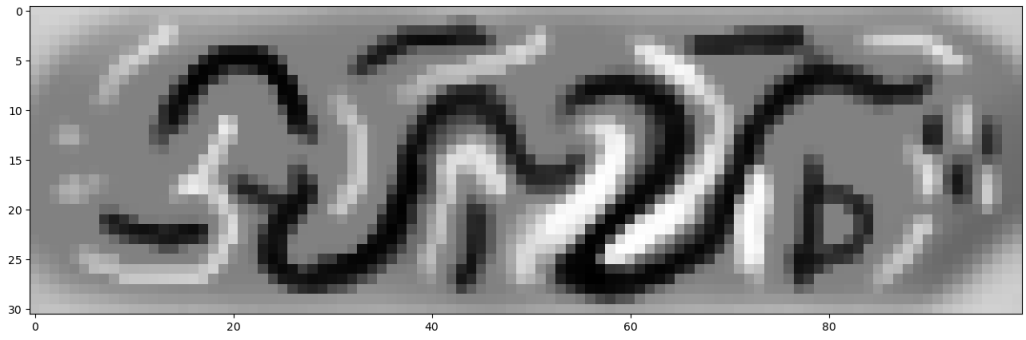
In the second part (*Report*), you are required to provide a report of maximum 2 pages describing the experiments you have run and analyzed to answer the *obligatory* questions, that is, questions marked by a star (*). The remaining questions are optional; however, you are invited to solve them because: (i) they are stepping-stones towards the obligatory questions; (ii) they are similar to the questions you will be asked at the exam. You will be evaluated only on your report. Review the documents on `coding tips` and `writing tips` for advices on how to tackle this part.

Exercise6 Dynamic Programming

Consider a robot searching an optimal path down the side of a mountain. Its aim is to find the fastest route to the root of the mountain, avoiding to traverse terrain that is too rough and uneven.

We will represent the side of the mountain as a *gridworld* made up by 31 rows and 100 columns (see image below). Some facts about the dynamics of the environment:

- The robot will start somewhere in the first column on the left (top of the mountain).
- The robot aims at reaching the rightmost column on the right (bottom of the mountain).
- Each square has a roughness index providing a reward signal of how fast the robot can move through that square.
- Because of the slope, at each timestep the robot can only choose whether to advance *straight to the right*, *diagonally to the top right*, or *diagonally to the bottom right*.
- If the robot attempt to move beyond the map in the top or bottom direction, it will fail to escape and just move straight ahead.



IMPORTANT: Pay attention to the encoding, as the indexing of the matrix may be counterintuitive. Notice that (0,0) is on the top-left corner!

1. Express this problem formally as a MDP.

The robot now wants to evaluate some alternative strategies relying on a model of the mountain it has acquired. Load the model `Mountain_one()` in `mountain.py` and implement an algorithm for policy evaluation.

2. Consider the policy π_{str} of heading straight ahead. Use policy evaluation to compute $v_{\pi_{\text{str}}}$.
3. (*) Run your policy evaluation ordering the states from left to right (from the top of the mountain to the bottom) or from right to left (from the bottom of the mountain to the top). How do the results differ?
4. (*) Consider the two polices evaluation in run with the different ordering of states. Did you learn something different between them?
5. Plot the trajectory followed by the robot according to π_{str} from $s_0 = (\mathtt{15}, \mathtt{0})$.
6. Plot the trajectory followed by the robot according to π_{str} from $s_0 = (\mathtt{3}, \mathtt{0})$.
7. If the agent were to follow the policy π_{str}, which starting position in the first column would be optimal?
8. Consider now a random policy π_{rnd} where each action is taken with uniform probability. Use policy evaluation to compute $v_{\pi_{\text{rnd}}}$.
9. Plot one sample trajectory that the robot could follow according to π_{rnd} from $s_0 = (\mathtt{15}, \mathtt{0})$.
10. (*) Did you learn something different between the policy evaluation for $v_{\pi_{\text{str}}}$ and $v_{\pi_{\text{rnd}}}$?
11. (*) Compare $v_{\pi_{\text{str}}}((\mathtt{15}, \mathtt{0}))$, $v_{\pi_{\text{str}}}((\mathtt{3}, \mathtt{0}))$, $v_{\pi_{\text{rnd}}}((\mathtt{15}, \mathtt{0}))$. If the robot can choose its initial position and its policy, which of the previous combination of initial state and policy has the highest value?
12. Reconsider the three policy evaluation that you have run. Is the issue of exploration affecting the final result?

Instead of just evaluating possible policies, the robot now considers trying to learn an optimal policy. Implement an algorithm for policy iteration.

13. Start from π_{rnd}; find an optimal policy using standard policy iteration where you alternate: (i) policy evaluation until the state values are stable (they change less than a ϵ threshold); and, (ii) policy improvement.
14. Plot the optimal trajectory.
15. Always start from π_{rnd}; find an optimal policy using truncated policy iteration where you alternate: (i) $k = 3$ steps of policy evaluation; and, (ii) policy improvement.
16. Plot the optimal trajectory.
17. Start at $s_0 = (\mathtt{15}, \mathtt{0})$ and π_{rnd}; find an optimal policy using value iteration. Implement value iteration both as a form of $k = 1$ truncated policy iteration and as a direct implementation of the Bellman optimality equation.
18. Plot the optimal trajectory.
19. (*) Are all the optimal solutions you computed above identical? Explain why they are or they are not identical.
20. (*) Describe the most efficient solution for policy iteration you have run. Was it standard policy iteration, truncated policy iteration, value iteration? How did you sort your states during the policy evaluation step?