COMP3702 Artificial Intelligence
Semester 2, 2022
Tutorial 7

Before you begin, please note:

- Tutorial exercises are provided to help you understand the materials discussed in class, and to improve your skills in solving AI problems.
- Tutorial exercises will not be graded. However, you are highly encouraged to do them for your own learning. Moreover, we hope you get the satisfaction from solving these problems.
- The skills you acquire in completing the tutorial exercises will help you complete the assignments.
- You'll get the best learning outcome when you try to solve these exercises on your own first (before your tutorial session), and use your tutorial session to ask about the difficulties you face when trying to solve this set of exercises.

Background

Recall from the lecture that in general, an MDP's **objective** is:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right].$$

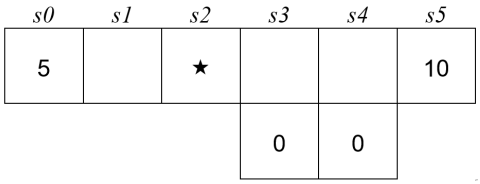
The **value function** of an MDP, $V^\pi(s)$, is the expected future cost of following an (arbitrary) policy, π, starting from state, s, given by:

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} P(s' \mid \pi(s), s) \left[R(s, \pi(s), s') + \gamma V^\pi(s')\right].$$

where the policy $\pi(s)$ determines that action taken in state s. Here we have dropped the time index, as it is redundant, but note that $a_t = \pi(s_t)$. Also note that $R(s, a) = \sum_{s'} P(s' \mid s, a) R(s, a, s')$.

Exercises

Exercise 7.1. Consider the gridworld below:



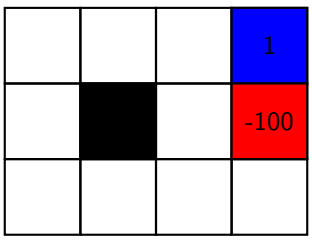
An agent is currently on grid cell s_2, as indicated by the star, and would like to collect the rewards that lie on both sides of it. If the agent is on a numbered square (0, 5 or 10), the instance terminates and the agent receives a reward equal to the number on the square. On any other (non-numbered) square, its available actions are to move Left and Right. Note that Up and Down are never available actions. If the agent is in a square with an adjacent square below it, it does not always move successfully: when the agent is in one of these squares and takes a move action, it will only succeed with probability p. With probability $1 - p$, the move action will fail and the agent will instead fall downwards into a trap. If the agent is not in a square with an adjacent space below, it will always move successfully.

a) Consider the policy π_R, which is to always move right when possible. For each state $s \in \{s_1, s_2, s_3, s_4\}$ in the diagram above, give the value function V^{π_R} in terms of $\gamma \in [0, 1]$ and $p \in [0, 1]$.

b) Consider the policy π_L, which is to always move left when possible. For each state $s \in \{s_1, s_2, s_3, s_4\}$ in the diagram above, give the value function V^{π_L} in terms of γ and p.

Example Gridworld domain

Consider the gridworld below:



States in this environment are the positions on the tiles. The world is bounded by a boundary wall, and there is one obstacle, at [1,1] (using python or zero indexing starting from the bottom left corner). In addition, there are two terminal states, indicated by the coloured squares.

Terminal states: In practice, we would say that the two coloured tiles are *terminal states* of the MDP. For mathematical convenience, we often prefer to deal with an infinite horizon MDP (see more below). To convert this model to an infinite horizon MDP, we will add an extra pseudo-state to the list of MDP states called *exited*, which is absorbing (the agent cannot leave it irrespective of its action, i.e. $T(exited, a, exited) = 1$ for all a).

Actions and Transitions: In this world, an agent can generally choose to move in four directions — *up*, *down*, *left* and *right* (which we might sometimes refer to as ^, v,< and >, respectively). However, the agent moves successfully with only $p = 0.8$, and moves perpendicular to its chosen direction with $p = 0.1$ in each perpendicular direction. If it hits a wall or obstacle, the agent stays where it is. In addition, once the agent arrives on a coloured square with a value, it has only one special action available to it; that is, to *exit* the environment.

Rewards: The values stated on the coloured squares are the reward for *exiting* the square and the environment, so the reward is not repeatedly earned; that is, $R([3, 2], exit) = 1$, and $R([3, 1], exit) = -100$. All other states have 0 reward.

Discount factor: $\gamma = 0.9$.

Exercises on Gridworld

For Exercises 7.2, 7.3 and 7.4, you will need to implement sections of `grid_world_starter.py`.

Exercise 7.2.

a) Implement the `attempt_move` and `get_transition_probabilities` methods in the `Grid` class in `grid_world_starter.py` according to their docstrings. The aim is to have functions that can be used on an arbitrary gridworld (i.e. do not hard-code your function just for this problem instance!).

b) What is the one-step probability of arriving in each state s' when starting from [0,0] for each a, i.e what is $P(s'|a, [0, 0])\ \forall\ a, s'$?

c) What is the one-step probability of arriving in state [1,0] from each state s after taking action a, i.e what is $P([1, 0]|a, s)\ \forall\ (a, s)$?

Exercise 7.3. Implement VI for this problem, using: $V[s] \leftarrow \max_a \sum_{s'} P(s' \mid s, a)\,(R(s, a, s') + \gamma V[s'])$.

Note that $R(s, a, s') = R(s)$ is the reward for landing on a square, which is non-zero for only the red and blue squares at [3,1] and [3,2], respectively.

a) What is the value function estimate after 4 iterations? What is the policy according to the value function estimate after 4 iterations?

b) What is the value function estimate after 10 iterations? What is the policy according to the value function estimate after 10 iterations?

c) Run value iteration until the maximum change in values is less than `EPSILON` (i.e. the values have 'converged'). At this point, what is the value function estimate? What is the policy according to the value function?

d) On each iteration, print the iteration number and the largest difference in the value function estimate for any state. What do you observe in the results?

Exercise 7.4.

a) Using an iterative approach to policy evaluation, implement PI for this problem, following:

- Set $\pi(s) => \ \forall\ s \in S$, and let $iter = 0$
- Repeat:
  1. Policy evaluation - Solve for $V^{\pi_i}(s)$ (or $Q^{\pi_i}(s,a)$):
  $$V^{\pi_i}(s) = \sum_{s' \in \mathcal{S}} P(s' \mid \pi_i(s), s) \left[R(s, \pi_i(s), s') + \gamma V^{\pi_i}(s')\right] \quad \forall s \in S$$
  2. Policy improvement - Update policy:
  $$\pi_{i+1}(s) \leftarrow \arg\max_a \sum_{s' \in \mathcal{S}} P(s' \mid a, s) \left[R(s, a, s') + \gamma V^{\pi_i}(s')\right]$$
  3. $iter = iter + 1$
- until $\pi_i(s) = \pi_{i-1}(s)$

b) Let $P^\pi \in R^{|S| \times |S|}$ be a matrix containing probabilities for each transition under some policy π, where:
$$P^\pi_{ij} = P(s_{t+1} = j \mid s_t = i, a_t = \pi(s_t))$$
Calculate the size of P^π in this gridworld, when the special pseudo-state *exited* is included.

c) Set the policy to move *right* everywhere, $\pi(s) => \ \forall\ s \in S$. Calculate the row of $P^>$ corresponding to an initial state at the bottom left corner, [0,0] of the gridworld.

d) Write a function that calculates the row (vector) of $P^>$ corresponding to an initial state at the bottom left corner, [0,0] of the gridworld for any action or deterministic $\pi([0,0])$. You should make use of methods you wrote in Exercise 7.2.

e) Turn this into a function that computes P^π for any deterministic π. Note that $P(exited|[3,1],a) = P(exited|[3,2],a) = P(exited|a, exited) = 1$ for all a, so the rows of P^π for these states s are all zeros except for a 1 corresponding to the transition to $s' = exited$.

f) Compute
$$V^> = (I - \gamma P^>)^{-1} r.$$
To do this, define r as the reward for landing on a square, which holds because $R(s,a,s') = R(s)$ for the red and blue squares at [3,1] and [3,2], respectively. (*Hint*: Rather than explicitly computing the matrix inverse, you may want to use `numpy.linalg.solve()` for large $|S|$. This implements the LU decomposition to solve for V^π using forward and backward substitution, so can result in a big speed-up.)

g) Replace your iterative approach to policy iteration with a linear algebra approach using what you have done over steps b-f.

h) How many iterations does PI take to converge? How long does each iteration take?

Exercise 7.5. **Question**: Derive V^π from the MDP objective function. That is, from:
$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right].$$
derive:
$$V^\pi(s) = \sum_{s' \in \mathcal{S}} P(s' \mid \pi(s), s) \left[R(s, \pi(s), s') + \gamma V^\pi(s')\right].$$