THE HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY

Department of Computer Science and Engineering

COMP4211: Machine Learning

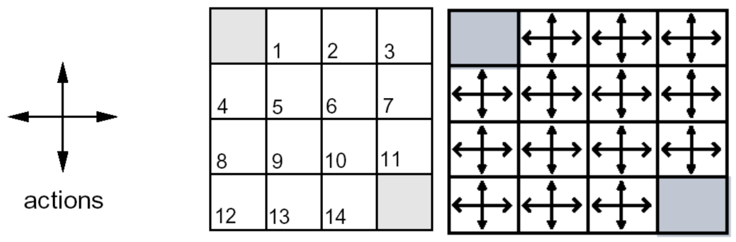
Fall 2019 Assignment 3

Due time and date: 11:59pm, Dec 10 (Tue), 2019.

IMPORTANT NOTES

- **All codes should be implemented in Python using the `numpy` module.**
- **Please make sure that your program output follows exactly the format specified in this assignment.**
- **Your grade will be based on the correctness, efficiency and clarity.**
- **Late submission: 25 marks will be deducted for every 24 hours after the deadline.**

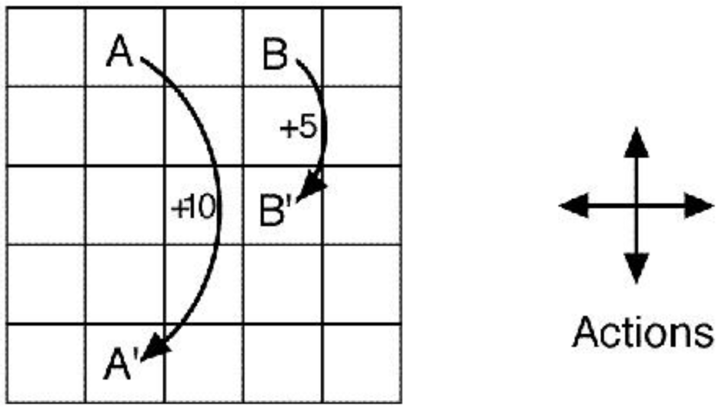
Q1. Consider the following environment with the shaded squares being terminal states. For each state, there are four actions (north, south, east, west). Actions that would take the agent off the grid leave state unchanged. For every action in every state, the reward is always -1. The diagram on the right shows a random policy. Let the discount factor γ be 1.



(a) By using the Bellman equation and the linear system solver provided in the course webpage, write a program to obtain the state value function. Your program should output a 4×4 matrix, in which each element represents the state corresponding value (to 2 decimal places).

(b) Instead of using a linear system solver as in part (a), write a program to obtain the state value function by iterative policy evaluation (with convergence tolerance $\Delta = 10^{-4}$).

Q2. Consider the environment shown in the following diagram.



For each state, there are four actions (north, south, east, west).

- At state A: Any action will move the agent to state A', and yields a reward of $+10$;
- At state B: Any action will move the agent to state B', and yields a reward of $+5$;
- Actions that takes the agent off the grid leave state unchanged but reward is -1;
- Any other action gets a reward of 0.

Set the discount factor $\gamma = 0.9$.

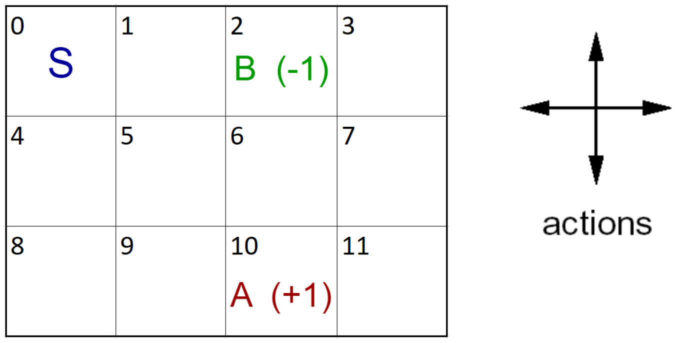
(a) Consider the random policy. Using your code in Q1(b), obtain the state value function (to 2 decimal places) by iterative policy evaluation (with convergence tolerance $\Delta = 10^{-4}$).

(b) Implement value iteration to obtain the **optimal** state value function (with convergence tolerance $\Delta = 10^{-4}$). Your program should output a 5×5 matrix, in which each element represents the corresponding optimal state value (to 2 decimal places).

Q3. Consider the following deterministic environment, which consists of 3×4 grid with 12 states. Almost all actions lead to zero reward, including actions leading to the outside of the grid, which does not change the state. The exception are actions, which lead agent on cells with states "A" and "B": agent gets positive reward $+1$ if it enters state "A" and negative reward -1 if it enters state "B". The episode does not stop on these cells. It stops after agent takes 5 steps. The initial state for the agent on the grid is called "S", from which it starts all trajectories.

(a) You are given the following trajectory τ_1 sampled from the experience replay buffer: starts from "S", then perform actions 'Down', 'Right', 'Right', 'Right', 'Up'. Using the Q-learning algorithm, with the `Q-table` initialized with zeros and discount factor $\gamma = 0.8$, show

- for each action a in τ_1, the experience in the form (s, a, r, s');
- the `Q-table` after learning on each experience (s, a, r, s') in τ_1. The `Q-table` should be shown as a 12×4 matrix (to one decimal place) with each row for one state, and each column for one



action (in the order ‘Up’, ‘Down’, ‘Left’, ‘Right’). You can update the `Q-table` by hand or by writing a program.

(b) You then continue learning from the following trajectories:

1. τ_2: starts from “S”, then perform actions ‘Down’, ‘Right’, ‘Up’, ‘Right’, ‘Down’;
2. τ_3: starts from “S”, then perform actions ‘Down’, ‘Right’, ‘Right’, ‘Down’, ‘Up’;
3. τ_4: starts from “S”, then perform actions ‘Down’, ‘Right’, ‘Left’, ‘Down’, ‘Right’.

Using the `Q-table` learned in part (a), show the learning progress by repeating part (a) for each of the above trajectories, one by one.

Submission Guidelines

The source code in Jupyter notebook (`.ipynb`) format should be emailed to **comp4211@hotmail.com** before the deadline. All the submitted files should be contained in a single zip package named like **A3_awangab_12345678.zip** (replace awangab with your ust account and 12345678 your student id). Note that all the codes should be compilable and well-commented (provide enough comments for each key line of code), otherwise you may lose some points if the code is very difficult to understand. Plagiarism will lead to zero point on this assignment.