ME491 : Programming Exercise 3

Due Date : Wednesday, 13 October 2021, 5 p.m.

1 Introduction

The goal of this programming exercise is to obtain an optimal policy and the corresponding value function using **on-policy model-free control**. You will use **SARSA(λ)**. Here are some general rules of the programming and results which have to be submitted:

• You have to use Python.

• The environment is a 6 by 6 grid world where the state is given as a 2D coordinate.

• The convention for the state is similar to 2D Matrix indexing. The top row is the 0th row and the left-most column is the 0th column. So, the most bottom-right element is encoded as [5, 5].

• The agent can move up, down, left, or right unless the wall that enclosing the grid world obstructs it. It's forbidden not to move at any time. Correspondingly, the agent should choose the action among the possible action candidates.

• Actions are encoded as the following rule - Up: 0, Down: 1, Left: 2, Right: 3.

• The objective is to find optimal policy when the goal state is set to [5, 5].

• A policy is deterministic and should be formulized as a numpy.ndarray having the shape of (6, 6). The look-up table of the state-action value functions should have the shape of (6, 6, 4).

– An example for the policy and the path:

```
[[3 3 1 3 2 2]
 [3 1 1 2 0 2]
 [0 3 1 1 1 2]
 [0 0 3 3 1 1]
 [0 2 2 3 1 1]
 [3 0 2 2 3 0]]
```

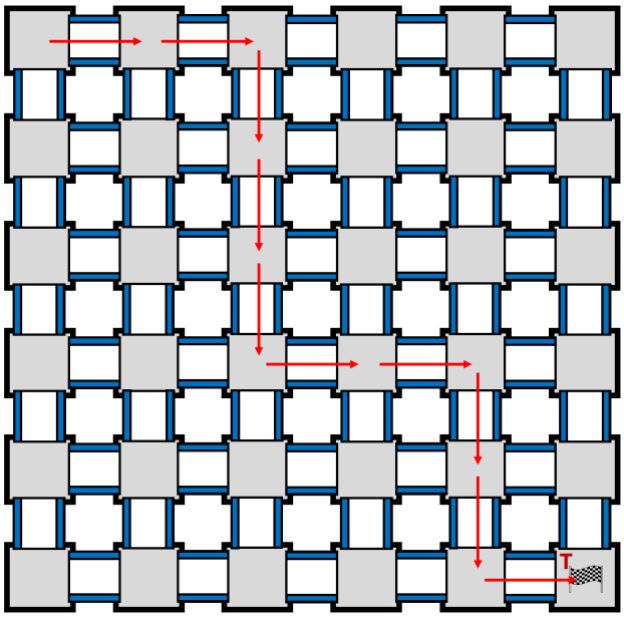

Figure1. Example path using obtained policy

2 Tasks

This exercise is consists of **2 tasks**. Both of the task will be based on the environment given in Figure2. Each grey box represents the state that the agent can reach. The red number that assigned on the bridge between two states represents the reward that will be given in that transition. The initial state is uniformly sampled from the all possible states except for the terminal state [5, 5]. Find the optimal policy in the given environment using different yet equivalent algorithms.

(1) Implement **forward-view SARSA(λ)** to find optimal policy.

(2) Implement **backward-view SARSA(λ)** to find optimal policy.

Remark

• Use ε-greedy policy to sample an action. Set **ε to 0.2**. Anneal ε by multiplying **0.99999** per each episode.

• Set discount factor **γ = 0.99**, learning rate **α = 0.01**, **λ = 0.8**.

• Train the agent until the accumulated number of episode reaches 1000000.

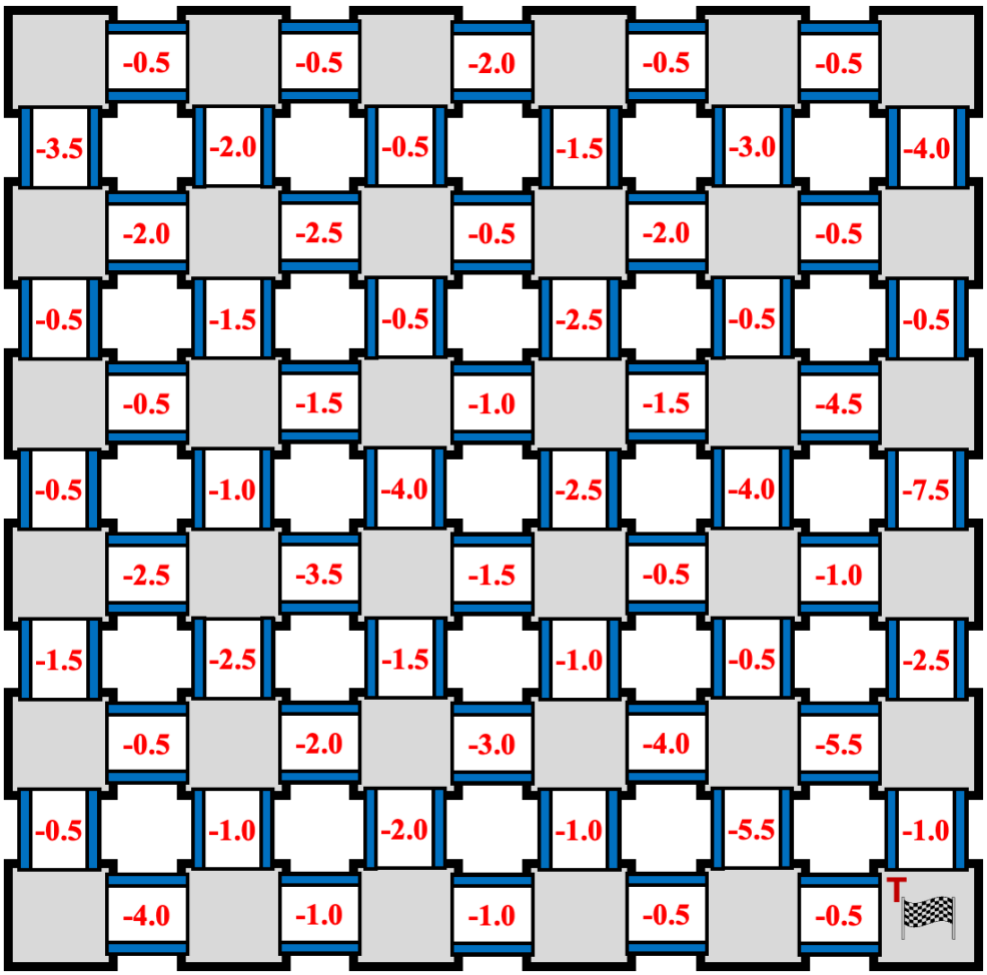

Figure2. Grid world for the tasks

3 Instruction

• A code template will be given.

• Do not change the environment.py file. You will only be graded by the code in the files named prob3_xxx.py.

• Although you are allowed to fix minor modifications to the code structure given, such as adding some auxiliary functions, but the main structure (e.g. name of functions or classes) should be remained so that the grader check your code properly.

• Use step and reset function of the GridWorld class for the agent to interact with the environment in the main loop.

• You will get scores for **optimal policy** and **the correctness of the script**. The value table will also be submitted, but will not significantly relate to grading.

Submission

Submit a zip file, consisting of 6 files

• The name of the zip file should be '(student number)_(name).zip', and the file names should be the same as below. An example of the final file is:

20200000_yourname.zip

- prob3_sarsa_lambda_forward.py
- prob3_sarsa_forward_policy.npy
- prob3_sarsa_forward_q_table.npy
- prob3_sarsa_lambda_backward.py
- prob3_sarsa_backward_policy.npy
- prob3_sarsa_backward_q_table.npy