Mid Term Project: Build an RL environment

Submission Due: Monday November 1st 11:59 pm

The purpose of this project is to prepare an RL environment/task that you will be using for evaluating some RL algorithms during the final project.

What are we looking for?

For this project, you should program a task or environment that your agent will use to learn during the final project in the class. To be consistent within all the homework and avoid a large variance of tasks, in this semester, we want you to focus on gridworld environments, a 2D environment divided into cells/grids which could include different features.

Task I – Familiarize yourself with Pygame

Pygame is a python library with tools that enable you to build a game/task/environment according to a Reinforcement Learning problem setup. If you decide to use Pygame, you should spend some time and familiarize yourself with the library. The best place to start is their documentation page at https://www.pygame.org/docs/ . You can also take a look at some of the existing games/tasks at https://www.pygame.org/tags/all

Task II – Build a Task/Game/Environment

One of the important goals in this project is to minimize the effort required to build the task for your final project. We are not looking for advanced visualization effects, instead, we are looking for ideas that can make the game more interesting and fun. Of course, your program should include all the features required by an RL algorithm.

When designing/implementing the task, you should think about the action set, state-space, reward function, transition function, algorithm you want to use to learn, and so on. You also should consider the computational cost for the learning process. For instance, a task that can run episodes without visualizing them is preferred over a task for which everything should be visualized.

In this task, you should:

1- Design and build a 'small' 2D gridworld environment. The size of this environment must be between 15x15 and 20x20 with less than 20% walls/obstacles. You should specify at least one goal and at least two different types of challenges (e.g., cells with negative rewards, blocked areas, stochastic regions). Note that you can get extra points for novel ideas that make the task more interesting, only if your algorithm can handle (solve) the challenge. Your agent must be able to move in 4 main directions, but you can add more actions according to the designed game (e.g., pick up, put down, open, close).
2- Extend the environment in previous step to a 'large' 2D gridworld environment. The size of this environment must be between 30x30 and 50x50 with less than 20% walls/obstacles. Similar to the previous part, you should specify at least one goal and at least 3 to 4 different types of challenges (e.g., cells with negative rewards, blocked areas, stochastic regions). Note that you can get extra points for novel ideas that make the task more interesting, only if your algorithm can handle (solve) the challenge. Your agent must be able to move in 4 main

directions, but you can add more actions according to the designed game (e.g., pick up, put down, open, close).

Requirements:
Environments must have two modes:

a) Keyboard interaction mode. This means, in this mode, you should monitor for key press events and allow a user to play the game.
b) Learning algorithm mode. In this mode, the user interface will be disabled and inputs from a reinforcement learning algorithm can be sent to be animated showing the actions of the agent in the environment. This can be done by implementing a function that can be called at any point during the learning (as well as in the end) and passing information (e.g., a trajectory) to be played and animated in the designed environment.

Template:
You will be provided with two template files: `gridworld_template.py` and `learner_template.py`. The `gridworld_template.py` file includes implementation of a simple 1D gridworld using the PyGame library. You should build upon this template, but you can look for ideas by searching through the existing implementations in the Pygame website. The `learner_template.py` file includes RL modules such as transition function, reward function, reinforcement learning algorithm among others. You should update and extend these modules according to your environments.

Task III – Random Walker or RL algorithm

In this step, you should use either a random walker or one of the RL algorithms you have implemented and apply it to both environments (small and large gridworlds) and show that (a) the learning loop runs without any error, and (b) you can use the outcome of the algorithm to animate the agent's behavior upon request (either during learning or at the end).

Deadline:

The deadline for submitting all the python files is on November 1st before 11:59pm. I suggest you start early and get feedback to have more time to incorporate comments and updates. I would like to remind you that the **mid-term project is worth 25% of your final grade**.

Evaluation: we will grade your submission according to the following table:

Item	COMP4600	COMP5500
Small gridworld	20	20
Small learner	20	20
Large gridworld	20	20
Large learner	20	20
Connections	10	10
Features/novelty	10	10

Submission: Submit all files (no more than 4 files) as **a single zip file** through Blackboard.