FIT5226 Project

Stage 1 - Tabular Q-Learning

This document describes Phase 1 of the FIT 5226 project, which will be conducted in 3 Phases overall. These have been described during the Seminar in Week 1 and are briefly summarized below. Each stage builds on the previous stages.

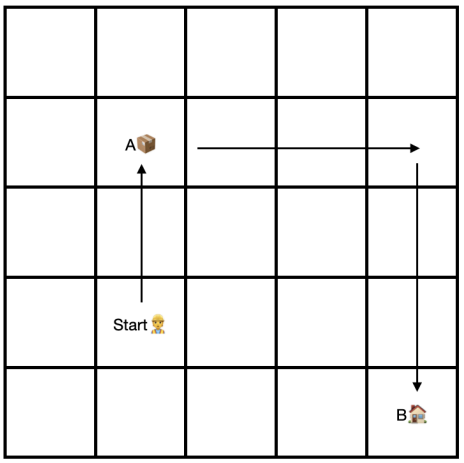
Assessment	Topic	Release	Due (Sunday)	Weight	Assessment mode
in-semester	Project Stage 1 (Table-based RL, single agent)	Wk3	Wk 5	10%	Group
in-semester	Project Stage 2 (Deep RL, single agent)	Wk 5	Wk 8	10%	Group
in-semester	Project Stage 3 (MARL)	Wk 9	Wk 12	55%	Mixed (comprising an individually conducted and assessed implementation and a group report giving a comparative evaluation)

This project and the mid semester quiz constitute your *entire* assessment for the unit. There is no exam.

Phase 1 (the current Phase) concerns *Tabular Q-Learning for Single Agents*. It is marked as a group assessment and is worth 10% of your overall mark.

Tasks for Stage 1

You will write code for a single agent in a square grid world of size *n* to learn a simple transport task. The agent's task is to pick up the item at location A and deliver it to a fixed target location B. A is not known in advance, ie. it varies each time the agent needs to solve the task. B is the bottom right corner of the grid at coordinates (*n*,*n*). The coordinates of A are part of the state information that the agent receives.



Your agent has four actions that it can execute: move north/move south/move west/move east (as in the examples given in the seminars). It starts at a random location. When it reaches location A it automatically picks up the item, when it reaches location B it automatically discharges the item. At this point, it has completed its task.

The agent is allowed to observe its own location, the location of A and whether it carries an item.

The task the agent has to learn is to pick up the load at A and deliver it to B taking as few steps as possible regardless of its (random) starting position.

1. **Reward Structure:** Design and describe a reward structure that will allow the agent to learn this task efficiently using reinforcement learning. A verbal description of the relevant parts will be sufficient (when you are interviewed about your solution).
2. **Environment:** Implement a grid world, in which the agent can move and execute its task. To make the task manageable, we use a 5x5 grid world. However, your code should be set up to work for any size (so use parameters for the size and for the target location). We simply choose a smaller grid here to limit memory and time requirements for the training. Note that you will have to integrate this with a visualization in the final task. It is highly advisable that you consider this for your code design right from the beginning.
3. **Learning:** Implement a table-based Q-Learning algorithm for this agent in Python as a Jupyter notebook. You are not allowed to use any reinforcement learning libraries for this, your Q-learning must be implemented "from scratch". You are, of course, allowed to use all modules in the standard distribution of Python (e.g. random). The only library you should need beyond this is Numpy (and later Matplotlib for the visualization). If you want to use any other additional libraries apart from Numpy and Matplotlib check with your tutor beforehand whether these are admissible.
4. **Training and Testing:** Train your Q-Learner. Devise a test procedure and metrics that you can use to show that your agent learns the task successfully and that it learns to solve the task independently of the location of A.

5. **Documentation:** Use the metrics you defined above to document in writing that your agent learns its task. Clearly explain how your test procedure and metrics work. You will also have to be ready to demonstrate this to your tutor and every group member will have to be ready to explain any aspect of your solution. Make sure to document all code appropriately.
6. **Visualisation:** Use the grid-world code from the first tutorial to visualise that and how your agent learns the task. For example, use the visualisation to show typical runs at different stages of the training. You are, of course, free to write your own visualisation code. The code from Laboratory 1 is provided only to make your task easier.

Submission Instructions

Submission will be via the Moodle platform. Detailed submission instructions will be published on Moodle in the Assignment section in Week 4.

Team Nominations

Teams are self-nominated and comprise 4 students. All teammates must be in the same lab class. Your team will need to nominate one contact person responsible for the submission.

All team members are individually and jointly responsible for all parts of the assignment. This means in particular, that *each* team member will have to be able to explain *every* aspect of the problem and solution.

Tell your TAs who you would like to be in a team with, and they will do their best to create teams based on that. Your TAs and the CE will have the final word on all team compositions.

If you do not attend and/or select a team the TA will place you in a team at their discretion. This will be final.

Use of Generative AI

You are allowed to use Generative AI to solve your assignment. If you decide to do so, you must treat the AI like another external author (as a non-authoritative author whom you mistrust, given how much content is made up by Chat GPT and similar AIs). It is entirely your own responsibility that the content is correct and you can only use generated content to the extent that you could use materials provided by an external author. The AI is not part of your project team. If you use code that an AI produced you must be able to fully explain every detail of that code.

Details will be given with the individual assessment items.

Any use of generative AI must be appropriately acknowledged (see Learn HQ).