University of Westminster
Department of Computer Science & Engineering

6ELEN018W – Applied Robotics Coursework (Semester 1)

Module leader	Dr D. Dracopoulos
Unit	Coursework
Weighting:	50%
Qualifying mark	30%
Description	
Learning Outcomes Covered in this Assignment:	- Design, implement and evaluate velocity controllers and trajectory controllers in robotics applications; - Evaluate the behaviour of complex robotic systems
Handed Out:	December 2024
Due Date	6/1/2025 13:00
Expected deliverables	1. Python code in the form of Jupyter notebook files for modelling and control (*.ipynb files) and all necessary input files to run the code 2. Report in PDF format
Method of Submission:	Online via Blackboard
Type of Feedback and Due Date:	Individual feedback via Blackboard within 3 weeks of submission **All marks will remain provisional until formally agreed by an Assessment Board.**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website :**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

6ELEN018W Applied Robotics - Assignment
Deadline 6/1/2025, 13:00

Dr Dimitris C. Dracopoulos
Email: d.dracopoulos@westminster.ac.uk

Description

Your task for this coursework is to control a robot driving a taxi in an unknown environment. The robot objective is to navigate the unknown environment in an optimum (or near optimum way) while interacting with it, receiving rewards and penalties. The robot uses intelligent control algorithms to control its actions and find its optimum actions.

You will be assessed for 3 parts (the details of which are described in the following sections of this specification):

1. Implement a reinforcement learning control approach which the robot uses to determine optimum actions, based on rewards and penalties received by the environment. The Q-learning algorithm will be used.
2. Implement a neural network approach which controls the robot and determines its next action. Python and the `sklearn` library will be used.
3. Write a technical report (no more than 5 pages) discussing your results with the inclusion of appropriate diagrams.

The Problem

It is highly desirable to have robots which are able to adapt and operate in unknown or changing environments and circumstances which they have not encountered before.

Here, we consider a robot driving a taxi in an unknown environment, trying to find the optimum path to a passenger location, pick up the passenger and then travel to a goal location and drop-off the passenger. The robot also controls the taxi in such a way so as to avoid collisions with obstacles. The discrete environment is a 5x5 maze shown in Figure 1.

There are four designated pick-up and drop-off locations (Red, Green, Yellow and Blue) in the 5x5 grid world. The taxi starts off at a random square and the passenger at one of the designated locations.

The goal for the robot is to drive the taxi to the passenger's location, pick up the passenger, move to the passenger's desired destination, and drop off the passenger. Once the passenger is dropped off, the episode ends.

The player receives positive rewards for successfully dropping-off the passenger at the correct location. Negative rewards for incorrect attempts to pick-up/drop-off passenger and for each step where another reward is not received.

A map of the maze in text format is shown below:

```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

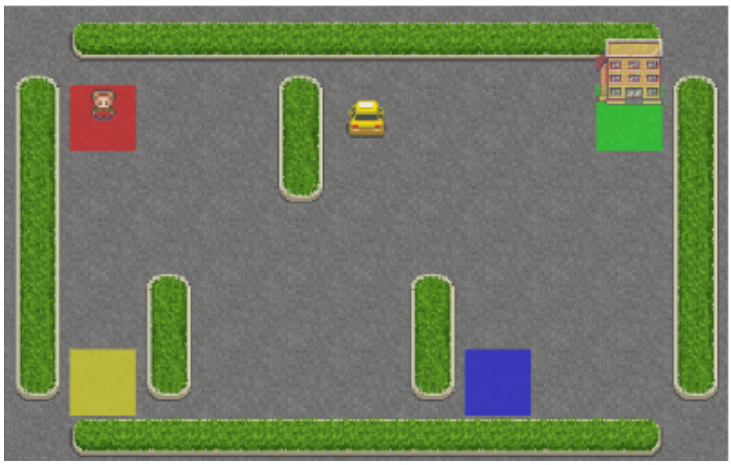

Figure 1: The taxi problem where a robot drives a taxi, in order to pick up a passenger located at a specific point and drive to drop-off the passenger at a specified target location. Different locations are used as starting points for the taxi, pick-up passenger points and drop-off passenger points in each episode.

At each time step, the action of the robot is indicated by a number in the range $[0, 5]$ which determines which direction to move the taxi or to pickup/drop off passengers:

- 0: Move south (down)
- 1: Move north (up)
- 2: Move east (right)
- 3: Move west (left)
- 4: Pickup passenger
- 5: Drop off passenger

There are 500 discrete states since there are 25 taxi positions, 5 possible locations of the passenger (including the case when the passenger is in the taxi), and 4 destination locations.

Note that there are 400 states that can actually be reached during an episode. The missing states correspond to situations in which the passenger is at the same location as their destination, as this typically signals the end of an episode. Four additional states can be observed right after a successful episodes, when both the passenger and the taxi are at the destination. This gives a total of 404 reachable discrete states.

The rewards for the robot at each time step are:

- -1 per step unless other reward is triggered.
- +20 delivering passenger.
- -10 executing "pickup" and "drop-off" actions illegally.

An action that results a noop (no operation), like moving into a wall, will incur the time step penalty.

For all the simulations and testing of your implemented algorithms, you should use the taxi problem implementation of the Gymnasium API RL. The same page contains more details about the implementation of the problem you are required to solve:

https://gymnasium.farama.org/environments/toy_text/taxi/

Instructions on how to install the environment and use a robot agent to interact with a Gymnasium environment can be found in the links below:

https://gymnasium.farama.org/introduction/basic_usage/

https://gymnasium.farama.org/introduction/train_agent/

The above instructions of how to install and interact with a Gymnasium virtual world environment (different than the taxi environment) have also been described in one of the tutorials/lectures of the module.

1 Reinforcement Learning Control of the Robot

Implement using Python the Q-learning control algorithm we have seen in the lectures and tutorials. You should implement this from scratch (based on the code given to you in the tutorials and extending it) and NOT use any library which implements the algorithm.

The robot learns the optimum Q values for determining the optimum action at every time step t while it interacts with the environment.

There are different optimal Q values for each configuration of the problem (passenger location, taxi starting location, drop-off location) and you should call your Q-algorithm implementation (e.g. a Python function) multiple times to evaluate these optimal Q values. You should save these to an appropriate Python data structure.

The robot taxi driver uses these learned Q values to determine the next optimal action at the current time t. This can be done by just looking ahead at the action that maximises Q at the current location.

In your report, you should discuss how well the algorithm of the robot works for the training and the unseen data using appropriate methodologies such as metrics and/or diagrams.

2 Neural Network Control of the Robot

Implement using Python and the `sklearn` library a multilayer perceptron using the backpropagation algorithm to determine the next control action of the robot taxi driver.

The robot uses the implemented neural network to decide (predict) which action to take next.

For training the neural network you should use optimum actions learned from the application of the Q-learning algorithm that you developed in the first part of this coursework. You should create a file with appropriate data (404 rows) corresponding to the form

```
(taxi_location, passenger_location, destination) -> optimum_action
```

and then split the data into training and testing datasets. This above file should be created by running your Q-learning algorithm for all the possible 404 configurations corresponding to all possible combinations of taxi locations, passenger locations and destinations.

You should split the data into appropriate sizes of training and testing sets so that the neural network of the robot is able to generalise best on the testing set.

In your report, you should discuss how well the neural network of the robot performs for the training and the unseen data. This should be done using appropriate methodologies such as metrics and diagrams.

3 Technical Report with Analysis of your Solution

You are expected to write a short report **in PDF format (no more than 5 pages)** which you discuss all your implementations for the 2 tasks.

You should discuss how you chose the split of training and testing data for the neural network as well as how you defined the rewards for the reinforcement learning approach.

Discuss in detail any of the choices made for the 2 different controllers and how they affect the performance of the control of the robot. You should include metrics, such as RMSE (root mean square error) and others, as well as any diagrams you think will demonstrate how well the controller performs.

Marking Scheme:

1. **Reinforcement Learning Controller**

   - Python implementation of the Q-learning controller: *20 marks* (correctness: 15 marks, efficiency: 3 marks, readability: 2 marks — The 15 marks for correctness will be awarded as:
     - 15 marks for a totally correct implementation
     - 11–14 marks for a very good implementation with minor omissions/mistakes
     - 6–10 marks for an implementation with major mistakes/omissions
     - 3–5 marks for a poor implementation
     - 0–2 marks for a sketchy or not very relevant implementation.

)

- Correct usage of the environment: *5 marks*
- Correct choice of the next optimal action based on calculated Q values: *10 marks*
  - 8–10 marks for optimal choice
  - 5–7 marks for an appropriate but not optimal choice
  - 3–4 marks for a poor action choice
  - 0–2 marks for an incorrect choice.

2. **Neural Network Controller**

- Python implementation: *10 marks*
  - 10 marks for a totally correct implementation
  - 7–9 marks for a very good implementation with minor omissions/mistakes
  - 4–6 marks for an implementation with major mistakes/omissions.
  - 0–3 marks for a poor implementation or not using the required libraries.
- Appropriate split of data set: *5 marks*
- Correct construction of training/testing data based on the Q-learning implementation: *10 marks*
  - Correct construction of data: 10 marks
  - 7–9 marks for minor mistakes/omissions
  - 4–6 marks for major mistakes/omissions
  - 0–3 marks for poor or totally incorrect construction of the data.
- Calculation of appropriate metrics: *7 marks*
  - Correct calculation of a number of different metrics: 7 marks
  - Correct calculation for inadequate choice of metrics: 4–6 marks.
  - Poor or incorrect calculation of metrics: 0–3 marks.
- Appropriate inputs and outputs and values of the parameters for the training, as given by the specification: *3 marks*

3. **Analysis Technical Report**

- Justification of choices made : *10 marks*
  - Excellent coverage of the justification: 9–10 marks.
  - Good but with minor omissions/mistakes justification 6–8 marks.
  - Major omissions/mistakes or insufficient coverage of justification: 3–5 marks.
  - Incorrect or no relevant justification: 0–2 marks.
- Metrics and diagrams explaining the results obtained: *10 marks*
  - Excellent choice of metrics and diagrams with appropriate explanation: 9–10 marks.
  - Limited choice of metrics/diagrams or without sufficient coverage of the explanation: 6–8 marks.
  - Incorrect explanation or poor choice of metrics and diagrams or diagrams not clear enough: 3–5 marks.
  - Irrelevant choice of metrics/diagrams or almost not existing discussion of them.

- Clarity of the technical report and completeness: *5 marks*
  - Clarity: 2.5 marks
  - Completeness: 2.5 marks
- Citations and references: *5 marks*
  - Proper citations: 3 marks.
  - Sufficient and relevant references: 2 marks.
- Penalty for exceeding 5 pages: *-20 marks*

Submission of assignments using a different method other than Blackboard will not be accepted and zero (0) marks will be awarded in such cases.

Deadline: Monday 6th of January 2025, 13:00.

Submission Instructions

Files to submit: All the Python Jupyter notebooks of your code, any required input files to run the code and the report in PDF format. **The notebook files should run on Jupyter lab and notebooks which run only in other environments will receive no marks**. All your files should be submitted in a single zip file.

You should submit via BlackBoard's Assignment functionality (do NOT use email, as email submissions will be ignored.), all the files described above. A single zip file with the name `wNNNNNNNN` (where `wNNNNNNN` is your university ID login name) containing all the above files should be submitted.

Note that Blackboard will allow to make a submission multiple times. Make sure before submitting (i.e. before pressing the `Submit` button), that all the files you want to submit are contained there (or in the zip file you submit).

In the case of more than one submissions, only your last submission before the deadline given to you will be marked, so make sure that all the files are included in the last submission attempt and the last attempt is before the coursework deadline.

Request to mark submissions which are earlier than the last submission before the given deadline will be ignored as it is your responsibility to make sure everything is included in your last submission.

The following describes how to submit your work via BlackBoard:

1. Access `https://learning.westminster.ac.uk` and login using your username and password (if either of those is not known to you, contact the Service Desk, tel: +44 (0) 207 915 5488 or log a call via `https://servicedesk.westminster.ac.uk`.).
2. Click on the module's name, `MODULE: 6ELEN018W.2023 APPLIED ROBOTICS` found under `My Modules & Courses`.
3. Click on the `Assessment->Submit Coursework->Coursework`.
4. Click on `View Assignment`.
5. Attach your zip file containing all of the required files, by using the `Browse` button.

6. Create a Word or PDF file with the following information:

    - *Comments*: Type your full name and your registration number, followed by:
      "I confirm that I understand what plagiarism is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."

7. Attach the file with the statement above.

8. Check that you have attached both the zip and the statement file.

9. Click the `Submit` button.

If Blackboard is unavailable before the deadline you must email the Registry at `studentcentre@westminster.ac.uk` with **cc:** to myself and your personal tutor before the deadline with a copy of the assignment, following the naming, title and comments conventions as given above and stating the time that you tried to access Blackboard. You are still expected to submit your assignment via Blackboard. Please keep checking Blackboard's availability at regular intervals up to and after the deadline for submission. You must submit your coursework through Blackboard as soon as you can after Blackboard becomes available again even if you have also emailed the coursework to the above recipients.