CM20252 & CM50263 Artificial Intelligence
Coursework 2
Özgür Şimşek

Date set: 18 March 2019

Date due: 31 March 2019, 20:00

Total marks: 100

This coursework will determine 5% of your mark for the unit.

Where to submit: CM20252 Moodle page

What to submit: Completed Jupyter notebook (.ipynb file)

You will be given a Jupyter notebook to work with. You must follow the instructions on this notebook and submit this particular notebook. Otherwise you will receive 0 marks.

Late submissions: We will follow the university policy on late submissions. Coursework submitted after the deadline will receive a maximum mark of 40 (out of 100). Coursework submitted after five working days will receive a mark of zero.

Feedback: Within two weeks of the submission deadline, your Jupyter file will be returned to you (via Moodle), showing the marks you received from each part of the coursework. You can get additional feedback from the unit leader or one of the tutors via appointment.

You are required to work individually. You are welcome to discuss ideas with others but you must design your own implementation and write your own code.

This coursework will be marked anonymously. Please do not include any identifying information on the files you submit.

Do not plagiarise. Plagiarism is a serious academic offence. For details on what it is and how to avoid it, please visit the following webpage:

http://www.bath.ac.uk/library/help/infoguides/plagiarism.html
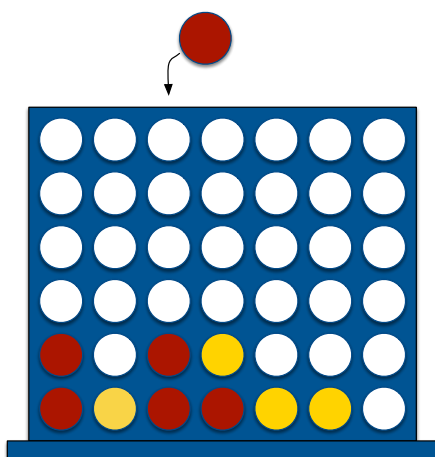
Connect Four

You will work on Connect Four, a small but relatively interesting game to try reinforcement learning on. Some of you may have seen this game before in other units. This time it will be about Q-learning.

Connect Four is a two-player zero-sum game played on a vertical grid, as shown in the figure. Each player is given disks of a particular colour. The players take turns dropping one of their disks from above the grid, into one of the columns. The piece falls straight down, occupying the lowest empty slot within the column. The objective of the

game is to be the first to form a horizontal, vertical, or diagonal line of four discs. If the grid fills up before either player achieves the objective, the game is a draw.

The original game is played on a 6×7 grid, as shown in the figure. The first player begins the game by dropping one of her discs into the centre column. You will work on a simpler version of the game, called *Connect Three*, where a player needs to connect only <u>three</u> pieces in order to win the game. In addition, you will use a grid smaller in size than the standard grid. Exact details are provided in the Jupyter file.

What you need to do

Apply Q-learning to learn an optimal policy when playing against an opponent who chooses moves uniformly at random[1]. Your opponent always plays first. Compare the performance of your policy to the performance of random play[1].

Use a reward function that assigns +1 for a win, –1 for a loss, and 0 for a draw. Use initial Q-values of zero. Choices of parameters α (step size) and ε are up to you. Experiment with different values to gain an understanding of how these parameters impact learning. You may change α and ε over time. It is also up to you to determine any other experimental parameters you will need, for example, how many games the agent should learn from.

You will be given some code that implements the game dynamics. This code is provided to you as a starting point. You may want to modify the existing code to suit your needs. <u>You will need to write additional functions.</u> You are also free to write your own code from scratch. If you modify the given code or write your own version, please note the following:

(1) You are responsible for implementing the game dynamics accurately.

(2) You must use a different file name for your code (i.e., anything other than "connect.py"). Otherwise our "connect.py" will override yours.

(3) You must submit all code required to execute your notebook.

You will notice that the existing code helps with the implementation of the game but not with the machine learning component of the coursework. This is intentional. You are expected to write your own code for the entirety of the agent's interaction with its environment in order to learn how it can best play this game.

Marking guideline:

In order to obtain full marks, you need to demonstrate an excellent understanding of Q-learning, with efficient learning of the optimal policy. Note that a more *efficient* method needs fewer observations to reach a certain level of play.

If you can demonstrate eventual learning of the optimal policy (but not necessarily efficiently), you will receive 65 marks or above, depending on how efficiently your agent learns.

---

[1] All columns that are not already full have equal probability of being selected.

If you can demonstrate some amount of learning, you will receive 40 marks or above, depending on how well your policy performs and how efficiently your agent learns.

Your code is expected to be well organised and readable. Otherwise, your marks may be lower than indicated in this guideline.

Additional details are provided in the Jupyter file.
<u>You must follow the instructions in the Jupyter file precisely.</u>

Two weeks of lab sessions are allocated for you to work on this coursework. Our tutors will be available during these lab sessions to answer your questions and to help you in any way they can. You can also consult the unit leader in her office hours, during the lectures, or by appointment.