Reinforcement Learning: Mountain Car++

6COM2007 Intelligent Adaptive Systems

Semester B 2024/2025

In this assignment, you will work on the classical Mountain Car problem, solve it with Reinforcement Learning and further adapt it to both solve the added challenge of stopping at a specific location. You will further support the agent in learning by creating a range of state space discretisations using a range of techniques to aid in learning and adapting. Reporting your results for the different implementations. And finally explaining and justifying your design choices.

You are allowed to use libraries, but if they solve the task for you, you will not receive marks. You are not allowed to use LLMs/GenAI in the creation of your code or report.

There are 50 marks to achieve, each translates to 1% of your overall module grade.

Example code is provided to get you started, you may adapt or forego this code, it will require extensive modifications either way. The problem statements in the form of the world design (world, goal state(s), force, dynamics, etc) may not be changed; if you think you need to make changes to these for any reason, please write Felix (f.riegler@herts.ac.uk) and get approval.

The tasks build on top of one another. If the state space changes, assume it changes for all of the next tasks going forward. For example after the second task all tasks should use the gradient space instead of the position space.

If you make changes to parameters related to the Q-Learning please make note of this in the task where you made the changes and why they were required.

Explanation tasks can usually be solved fully in below 5 sentences. Depends on the task, what you previously explained, and your results. Please try and be specific and to the point. I do not mind additional explanation, especially if interesting, but at some point I will stop reading if you go well over this :)

1 Implementing Q Learning - 14 marks

The first task is to implement basic Q-Learning on the classic mountain car problem.
This is an episodic problem, the episode ends once the agent reaches the top of the mountain (goal) or fails by going into the other side.

- Implement Q learning in the code. Your agent knows its current position. [4marks]
- Briefly explain your reward function. [1mark]
- Explain how you discretised the state space using one dimensional binning. [2marks]
- Explain your choice in α and ϵ and their decay over time. [3marks]
  As well as your choice in γ. [1mark]
- Plot how many actions every episode has over time while learning. [1mark]
- Run this 10 times, plot all 10 runs in the same figure. [2marks]

2 Gradient state space instead of GPS - 7 marks

The task remains the same but with a change to the state space. Our car no longer knows its position, it only knows its current slope given by the gradient (already implemented in the example code).

- Solve the task above. [2marks]
- Explain how you discretised the the new state space using one dimensional binning. [2marks]

- Plot how many actions every episode has over time while learning. Run this 10 times, plot all 10 runs in the same figure. [2*0.5marks]
- Discuss the difference in results to the previous task. [2marks]

3 Adding a garage/parking spot - 7 marks

In this extension a parking spot in the second valley is added (see code). The episode now ends after reaching the wrong hill or after decelerating and coming to a rest in the parking spot.
The state space is the same as in the last task.

- Solve the task above. [2marks]
- Explain how you discretised the the new state space using one dimensional binning. [2marks]
- Plot how many actions every episode has over time while learning. Run this 10 times, plot all 10 runs in the same figure. [2*0.5marks]
- Discuss the difference in results to all the previous tasks. [2marks]

4 Grid Binning using two colours/sections - 10 marks

Extend the state space from only the gradient to the gradient and one of two colours.
All states need a colour assigned, there is exactly one switch between them (i.e. all states left of x are green, all states right are red).
Think about how to choose x and why this makes sense for the learning task.

- Solve the task above. [2marks]
- Explain how you discretised the new state space. [1marks]
- Justify your choice of x, feel free to contrast this with failed choices. [4marks]
- Plot how many actions every episode has over time while learning. Run this 10 times, plot all 10 runs in the same figure. [2*0.5marks]

- Discuss the difference in results to all the previous tasks. [2marks]

5 Additional colours to enhance the state space - 6 marks

Same as task above, but now you may use as many colours as you want, justify the number. AND your colours may start and end as you see fit.

- Justify your number and choice of colours over the x-dimension. [3marks]
- Plot how many actions every episode has over time while learning. Run this 10 times, plot all 10 runs in the same figure. [2*0.5marks]
- Discuss the difference in results to all the previous tasks. [2marks]

6 Coarse Tiling - 6 marks

Same as task above, but now each state may contain multiple colours.

- Justify your number and choice of colours over the x-dimension, in particular the overlaps. [3marks]
- Plot how many actions every episode has over time while learning. Run this 10 times, plot all 10 runs in the same figure. [2*0.5marks]
- Discuss the difference in results to the non-coarse case. [2marks]

Bonus

You may extend this task further for bonus points. Explain your rationale, what you did, and analyse the results.

Feel free to take inspiration from other topics discussed during this module or other modules. Things like, adding noise over steering and/or sensors and using Kalman-Filters or PID. Or using curiosity based exploration around entropy. Or adding a controller with state transitions and learning on the state space of the controller instead of directly controlling the car. Or changing the world an exploring the borders of physics and learning. Or .. :)

Good and interesting implementations would usually warrant up to 5 bonus points, but truly exceptional and/or extensive work might go up to 10.

Submission requirements

The work must be your own. You may of course collaborate with peers but the work handed in must be distinctly yours. The following two sets of documents must be submitted on StudyNet/Canvas:

(a) A *.zip* archive containing your commented code. One file for each task. You may use additional code files to reduce redundancy. Each task-file should execute itself when run.

(b) A report in PDF format providing the explanations, numbers, and figures requested in the tasks given below.

You do not specifically get marks for comments, but where code is required and not readable marks might be deducted based on unreadability. Assume the reader of your code is one of the better programmers in your cohort.