4. Exercise Sheet

As it was the case on the last exercise sheet you can solve the programming exercise in pairs. We tried to reduce the time needed for the task by pre-implementing some steps.

Exercise 1 Regret - Understanding

In the lecture we reviewed multiple methods for action selections and their regret in different Multi-armed Bandit scenarios. Explicitly we discussed Random selection, Greedy selection, ϵ Greedy selection and Soft-Max selection.

a) Explain why we measure regret, why we want to minimize it, and how the different functions approach this problem.

b) Think about at least one Multi-armed Bandit scenario per method, where the method should perform very well and one were it will have trouble in finding the best solution. In case this cannot be fixed with a certain setup of bandits explain which problems should arise during the initialization of each method.

Hint: A Multi-armed Bandit scenario is characterized by a set of bandits, which each have (not necessarily) different chances of winning, e.g. the scenario (0.1, 0.1, 0.3) consists of three bandits, of which choosing either of the first two would yield a win-rate of 10% and choosing the third one yields a win-rate of 30%.

Hint 2: You can use the program of the third task to search for suitable scenarios or validate your thoughts on the second part of this exercise. Just open *plotbandits.py* and add a new scenario to the function *get_scenario*. Executing *banditsimulation.py* will by default show you the result of the lecture scenarios. Add the id of your newly created scenario to the for loop to see the respective result.

Exercise 2 Regret and Bounds - Understanding

Imagine that you have to develop an agent for a game that lasts for at most T (e.g. T =1000) time steps. You have a model of this game and you are using Flat Monte Carlo.

a) When is exploration/exploitation more important and why?

b) How can you vary the agent's behaviour towards exploration / exploitation depending on the time, when one of these is more important?

Exercise 3 Regret - Implementation

This task can be solved in pairs!

Due to various drawbacks of the methods Random selection, Greedy selection, ε-Greedy selection, and Soft-Max, we discussed how the concepts of exploration and exploitation can help us in coming up with another method.

a) Download the Python package hosted at:

`http://www.is.ovgu.de/is_media/MultiArmedBandit.zip`

b) Implement decaying ε-Greedy selection in the file *banditsimulation.py*

c) Implement Upper Confidence Bounds in the file *banditsimulation.py*

d) Test your programmed action selection methods on the same scenarios as in the lecture and describe your observation. You can conveniently access the scenarios executing *banditsimulation.py*

Hint: The provided BanditStrategy class already stores the number of trials (*bandit.trials*) and the number of wins per bandit (*bandit.wins*). Both implementations will only consist of a few lines of code (< 10).

Exercise 4 Monte Carlo Tree Search (MCTS) - Understanding

a) Describe and sketch the basic structure of MCTS and the process of building such a tree search. Therefore, summarize the 4 basic phases of MCTS and specify the desired outcome of the search process.

b) Compare the Tree and Default Policy and name their differences.

Exercise 5 Monte Carlo Tree Search - Application

Based on the example Tic-Tac-Toe explain the following elements of MCTS:

a) What is stored in each node of the tree?

b) How can we transition between nodes?

c) What is the maximal length of a simulation?

d) What is the maximal number of options?

e) How do we choose the node to be expanded?

f) What is done during the simulation?

g) What do we update after the simulations are done?

h) Which action do we execute after a reasonable number of simulations ?