Advanced Topics in Artificial Intelligence

T-720-ATAI-2016

EXERCISE 1 (Getting started)

In this exercise you will download and install the **TEAL** *(Tool for Evaluating Agents and Learners)* and learn to think about what it means for an agent to solve a task.

Download TEAL

Download TEAL from https://github.com/ThrosturX/rl-mouse/tree/exercises. Check out the documentation to get it running. Use the 'exercises' branch.

Make sure you have Python (at least 2.7) is installed on your machine.

Quick note 1: It is assumed you have the skills to get everything needed for these exercises up and running without assistance; no assistance can be guaranteed for this from the course instructors.

Implement Solutions

You will complete the exercises by implementing solutions that you should place in the 'solutions' folder. To evaluate a solution, execute `teal.py` with the appropriate arguments. In most cases, `'--grid_size N'` will suffice. Set N to 8 unless another grid size is specified.

Quick note 1: For many exercises it may suffice to change the agent instantiation.

Quick note 2: It can take a few minutes to evaluate solutions, depending on the exercise and your implementation.

The mouse, the cheese and the trap

In the first few exercises we will use the following task-environment: a mouse (the agent) is looking for cheese while trying to avoid stepping on any traps. The mouse lives in a circular grid-world, which means that if the mouse seems to walk off the edge, he will emerge on the other side. The mouse can move forward, left or right, and will face in the direction of the last movement. The mouse, cheese and trap all take up a single, randomly selected cell initially. When the mouse moves onto a cell with the cheese/trap, he will receive a positive/negative reward, and the cheese/trap is moved. The mouse will also receive a severe negative reward if he doesn't find the cheese after a large amount of time, in which case the locations of cheese and trap are also changed.

During the course of the exercises we will experiment with different grid sizes, and sensors, actuators and learning algorithms for the agent.

Solving the task yourself

In later exercises we will have the mouse learn how best to find the cheese and avoid the trap. In this exercise however, the mouse will not learn anything. It is up to you to program the mouse's behavior or "policy". This means that for every possible input, you have to specify the mouse's action. The maximum score for the mouse is 400 if you followed the instructions correctly.

Make a solution that creates a DeterministicMouseAgent with the actions `['left','forward','right']` in the "solutions" directory (you can use `ex1_template.py` as a template). Leave the `train` function empty. You only need to edit the

The mouse gets four inputs: the x,y-coordinates of the cheese and the x,y-coordinates of the trap. All coordinates are relative to the mouse and >= 0. In other words: they specify how far the cheese/trap is in front and to the right of the mouse, but you should keep in mind that the grid is circular.

A. Implement a policy that always moves forward and call `teal.py <solution-file> --grid_size 3 --max_epochs 5 --seed <your-kennitala>`. Take a look at the graph and the end summary. What is the average accumulated reward? (If you use a seed of 0 or omit the argument, you will get a slightly different result each time. Use your kennitala as the seed for submitting your final answer.)

B. Now implement a policy that always moves left, and evaluate it. What is the average accumulated reward?

C. Which one is better? What did you expect and why? If the outcome wasn't as you expected, try increasing the --max_epochs argument and/or run the test a few times without a random seed and see if you get what you expected.

D. Now implement a policy that gets the maximum score of 400 on any (reasonable) grid size. You could try to implement it first for small grids that you know the dimensions of (e.g. 3x3, 4x4), and generalize it from there.

E. A reward of 400 means that the mouse always finds the cheese and is never caught in a trap. It doesn't mean that the mouse took the best route. Try to create a policy that also takes the best route. This will increase the "ratio" and "performance" in the graph. A ratio of 1 means you're taking the optimal route. Test this on somewhat larger grids as well (e.g. 8x8).

Submit your solutions

Rename your solution file to "ex1_" followed by your name ".py" and put your answers to the above questions (A, B and C) in a file with the same name, but a ".txt" extension. Please write your full name and kennitala into both files (as a comment in the Python file). Submit them in MySchool.