**Exercise #3 – MDP and Reinforcement Learning**

Coffee World

# Problem Description

You would like to drink coffee. However, the coffee machine is in a different room and you do not want to go and get it yourself. So, you would like to build a robot and train it to get you coffee. The floor, unfortunately, is slippery, and so the robot may spill the coffee while going from the coffee machine to you.

You goal: build a robot that can plan its path intelligently so as to maximize the expected utility of drinking coffee. Note that while drinking hot coffee is worth 100 points to you, every step the robot moves cools down the coffee, effectively reducing the number of points by half. So, if the robot performs two steps from the coffee machine and gives you the coffee, you get 25 points.

Some details:

1. The reward for drinking a hot cup of coffee is 100. The reduced reward due to the time spent by the robot getting you the coffee is handled by a discount factor of 0.5.
2. The probability of spilling the coffee depends on the specific location. For example, the probability of spilling the coffee at location (1,1) may be 0.5 while the probability of spilling the coffee at location (2,1) may be 0.25.

# Task #0: IDs [10 points]

Before starting work on the assignment, find the class called ‘IDs’. Replace the empty strings in fields ‘id1’ and ‘id2’ with your ID numbers.

# Task #1: Markov Decision Process and Value Iteration [40 points]

In this task, the agent knows the probability of spilling the coffee in each grid cell. So, it can plan before acting, and then choose the best action. To this end, you will build an agent that runs the Value Iteration algorithm, and then chooses the optimal action.

Almost everything is already implemented for you in the *ValueIterationAgent* class. You must implement the method in the *BelmanUpdate* class:

* getNewV(state,environment, discountFactor, vValues)

This method computes the new V value of a state, using the current V values of all states. As a reminder, the Bellman update rule is as follows:

Where is the discount factor, which is already given to you in the code (see the code).

Also note that in the summation part, you only need to sum the states you may reach by performing a legal action in the current state.

To solve this task, it is strongly recommended that you observe the code in *ValueIterationAgent.chooseAction(state)* method in *ValueIterationAgent*.

To check your implementation, you can run the Runner class and observe the average discounted reward you get by running your agent on a randomly generated problem, and on a pre-generated problem. These will be shown in the lines starting with "MDP:". The expected average for the pre-generated problem is 31.64.

# Task #2: Reinforcement Learning with Q-Learning [40 points]

In this task, the agent does not know the probability of spilling the coffee in each grid cell and it needs to learn this over time. To this end, you will build an agent that runs Q learning and implements the -greedy action policy.

Almost everything is already implemented for you in the *EpsilonGreedyAgent* class. You must implement both methods in the *QValueUpdate* class:

* newQValue(oldQValue, reward, nextStateMaxQ, discountFactor, learningRate)

This method computes the Q value update rule. As a reminder, this rule is:

Where is the learning rate and is the discount factor.

* chooseAction(state, epsilon, environment, agent)

This method chooses which action to perform in the given state. You are supposed to implement the -greedy method, which means with probability you choose a random action and with probability you choose the action that maximize the Q value.

To check your implementation, you can run the Runner class and observe the average discounted reward you get by running your agent on a randomly generated problem. These will be shown in the lines starting with "RL:". If implemented correctly, you should expect to see the average utility rise and converge through many iterations.

**Scoring and submission instructions.**

You must only hand in the files ‘IDs.java’, ‘BellmanUpdate.java’ and ‘QValueUpdate,java’. Put all three files in the topmost hierarchy of a zip folder. The name of the zip folder will be your ID numbers, separated by a ‘\_’, like ‘123456789\_987654321.zip’. Assignments must be submitted through Moodle, by only one person from every pair.

Your code will be compiled using java 1.8, so make sure that your code is compatible with it.

The assignment is due by 16.1.20 at 23:59.

Good luck!