****

**Department of Software and Information Systems Engineering**

**Artificial Intelligence**

**(372-1-3502)**

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

**Due date: 15.1.2019**

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.

To complete this assignment, you have to download the code from the following Git repository

# https://github.com/ronistern/AAI-RL.git

# Task #1: Reinforcement learning with Q-Learning

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 only need to fill in details in two methods:

* computeNewQValue(state, action, newState, reward)

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

Where is the learning rate and is the discount factor. Both of these values are already given to you in the code (see the code).

* chooseAction(state)

This method chooses which action to perform in the given state. You are suppose 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:".

# Task #2: Markov decision process and Value Iteration

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 choose the optimal action.

Almost everything is already implemented for you in the *ValueIterationAgent* class. You only need to fill in details in one method:

* bellmanUpdate(state)

This method computes the new V value or 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 the *chooseAction(state)* method that is already properly implemented.

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 "MDP:".

**Scoring and submission instructions.**

Solving one of the tasks will earn you 85 points. If you wish to earn 100 points, you need to solve both tasks. Exact submission instructions will be sent via Moodle.