# CSEN 166 Artificial Intelligence

## Lab Assignment #5 Reinforcement Learning

**This is a GROUP assignment.** You will download **rl.zip** and modify the functions in **“class QLearningAgent(ReinforcementAgent):”** of **qlearningAgents.py**

**Task A:** Implement a Q-learning agent to walk in the grid world we saw in class. The agent learns by trial and error from interactions with the environment through its update(state, action, nextState, reward) method. A stub of a Q-learner is specified in QLearningAgent in qlearningAgents.py, and you can select it with the option '-a q'.

You will implement the update, computeValueFromQValues, getQValue, and computeActionFromQValues functions in qlearningAgents.py.

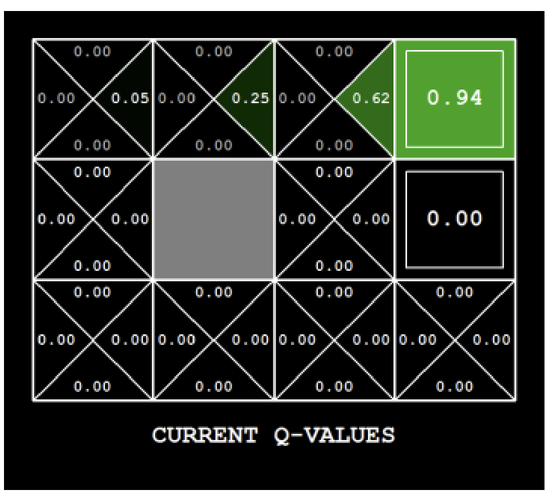
*Note:* For computeActionFromQValues, you should break ties randomly for better behavior. The random.choice() function will help. In a particular state, actions that your agent *hasn't* seen before still have a Q-value, specifically a Q-value of zero.

*Important:* Make sure that in your computeValueFromQValues and computeActionFromQValues functions, you only access Q values by calling getQValue.

***Task A – Experiment 1*** With the Q-learning update in place, you can watch your Q-learner learn under manual control, using the keyboard:

python gridworld.py -a q -k 4 -m --noise 0.0

Note: -k will control the number of episodes your agent gets to learn, -m means you manually control Pacman's actions, and --noise 0.0 means there is no randomness (it's a deterministic game). Please manually steer Pacman north and then east along the optimal path for **four episodes**, and you should see the following updated Q-values:



Hint:

- In computeValueFromQValues and computeActionFromQValues, the available actions shall be obtained by: self.getLegalActions(state)
- Watch how the agent learns about the state it was just in, not the one it moves to.

***Task A – Experiment 2*** Run the command:

python autograder.py -q q3

Your code is expected to pass all four test_cases of the above command.

**Task B:** Keep the modified code in Task A. Now, you will continue to modify the "def getAction(self, state):" function in the same qlearningAgents.py file. This function computes the action to take in the current state. There is a parameter self.epsilon, which is a probability. With probability self.epsilon, the agent should take a random action of all legal actions available in that state. With probability 1- self.epsilon, the agent will take the best policy action obtained by the computeActionFromQValues function.

***Task B – Experiment 1*** Run the following command, and observe how the Q-learner updates the Q-state values for 100 episodes.

python gridworld.py -a q -k 100

***Task B – Experiment 2*** Run the following command and observe how the crawler bot learns to move to the right. You can decrease the "Step Delay" to 0.0125 by pressing the top-left "-" button, so that you will see the displayed results sooner.

python crawler.py

***Task B – Experiment 3*** Run the following autograder, and you code should pass all four test_cases.

python autograder.py -q q4

**Submission:**

1. Submit a pdf file to Camino (the format of the file is similar to Lab4_submission_sample.pdf. Also include a screen shot/picture of the results you get for each experiment. For instance, for the crawler bot experiment, include a picture of the crawler bot when it's in the middle of the scene.).

2. Submit all source code needed (with qlearningAgents.py modified by you) to generate the results of **all Experiments in Tasks A and B** as a .zip file to Camino. We will test run your submitted code, so make sure it works.

**Demonstration:** you will demonstrate all experiments in tasks A and B to the TA, and explain how you implemented the following functions in your qlearningAgents.py code:

- getQValue
- computeValueFromQValues
- computeActionFromQValues
- update
- getAction

These functions are essential components of your algorithm, and you will also explain them in your lab report.

**Grading:** All demonstrations must be done before the lab report is due. The grade depends on both the correctness of the demonstration and submitted report and code.