COEN 166 Artificial Intelligence

Lab 5

Divya Syal: 1594122, Marianne Yamazaki Dorr: 1606975: , Suvass Ravala: 1593088

Lab 5 Contributions of each group member (in percentage): Each did 33%

Function 1: getQValue

paste your code:

```
    def getQValue(self, state, action):
    """
      Returns Q(state,action)
      Should return 0.0 if we have never seen a state
      or the Q node value otherwise
    """
    "*** YOUR CODE HERE ***"

    if (state,action) in self.qvalues: # (state,action) is the key; (s,a) is the key (s,a):
0.5
      # return the associated q-value
      return self.qvalues[(state, action)]

    else:

      # update the dictionary self.qvalues with this unseen (s,a) and initialize its
value as 0.0
      # self.qvalues[(s,a)]=0.0
      self.qvalues[(state, action)] = 0.0
      # return this q-value
      return self.qvalues[(state, action)]
```

In this function, we will return the QValue associated with the state, if the state exists then we will return the QValue, if not then we will create a qValue for that state, and set it to 0.

Function 2: computeValueFromQValues

paste your code (any other function that you defined or modified): def function_name(arguments):

```
def computeValueFromQValues(self, state):
    """
      Returns max_action Q(state,action) # which is V(state)
      where the max is over legal actions.  Note that if
      there are no legal actions, which is the case at the
      terminal state, you should return a value of 0.0.
    """
    actionList = []
    actionList = self.getLegalActions(state)
    max_value = 0.0
    if actionList:
        #for the first action
        max_value = self.getQValue(state, actionList[0])
        for action in actionList:
            if self.getQValue(state, action) > max_value:
                max_value = self.getQValue(state, action)

        return max_value
    else:
        return 0.0
```

…

In this function, we will create an actionList by getting all the legal states. If the action lists exist then we would go through all of the actions and find the max value of the actions by comparing the actions. If it doesn't exist then it returns 0.

Function 3: computeActionFromQValues

paste your code (any other function that you defined or modified): def function_name(arguments):

```
def computeActionFromQValues(self, state):
    """
      Compute the best action to take in a state.  Note that if there
      are no legal actions, which is the case at the terminal state,
      you should return None.
    """

    actionList = []
    actionList = self.getLegalActions(state)
    max_value = self.computeValueFromQValues(state)

    max_action = None
    max_actionList = []
    if actionList:
        for action in actionList:
            if (self.getQValue(state, action) == max_value):
                    max_actionList.append(action)

        max_action = random.choice(max_actionList)
        return max_action
    else:
        return max_action
```

In this function, we will first create a list of the legal actions. Then we will create a maxActionList to store all the values with max actions. We will run through the list of actions and find the max actions and if it is the max action then we will append it to the list. Then we will return a random action out of the list of maxActions. If the actionList does not exist then we will just return the max action.

Function 4: update

paste your code (any other function that you defined or modified): def function_name(arguments):

```
    def update(self, state, action, nextState, reward):
      """
        The parent class calls this to observe a
        state = action => nextState and reward transition.
        You should do your Q-Value update here
        NOTE: You should never call this function,
        it will be called on your behalf
      """

      # self.qvalues[(state,action)] = ...
      sample = reward + self.discount*self.computeValueFromQValues(nextState)
      self.qvalues[(state, action)] = (1 - self.alpha)*self.getQValue(state,
action)+self.alpha*sample
      #self.qvalues[(state, action)] =self.getQValue(state, action) + self.alpha*(sample -
self.getQValue(state, action))
```

In this function, we will calculate the q value by using the equation provided in class. First we calculate the sample, then we set the QValue as (1 - self.alpha) * self.getQValue(state, action)+self.alpha*sample.

Function 5: getAction

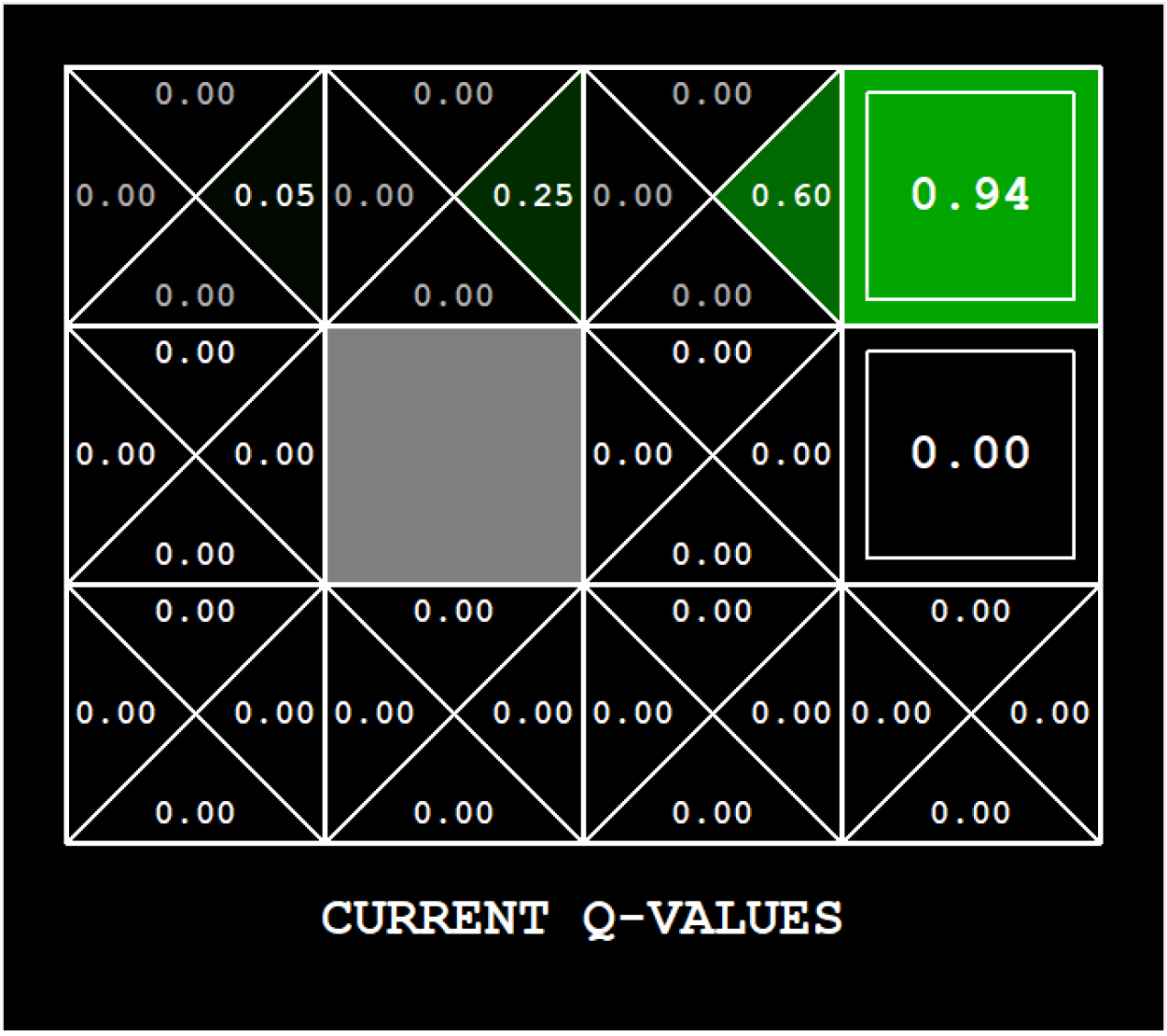
paste your code (any other function that you defined or modified): def function_name(arguments):

```
def getAction(self, state):
    """
      Compute the action to take in the current state.  With
      probability self.epsilon, we should take a random action and
      take the best policy action otherwise.  Note that if there are
      no legal actions, which is the case at the terminal state, you
      should choose None as the action.
      HINT: You might want to use util.flipCoin(prob)
      HINT: To pick randomly from a list, use random.choice(list)
    """
    # Pick Action
    legalActions = self.getLegalActions(state)
    action = None
    "*** YOUR CODE HERE ***"
    prob = self.epsilon
    if legalActions:
        chance = util.flipCoin(prob)
        if chance:
            action = random.choice(legalActions)
        else:
            action = self.computeActionFromQValues(state)

    return action
```

In this function, we get all the legalActions. If there are no legal actions, then we just return no action. If the legalAction exists then we calculate true/false value by using the epsilon probability. If it is true then we return a randomAction in the legal Actions. If it is false then we return the best action.

Experiments:

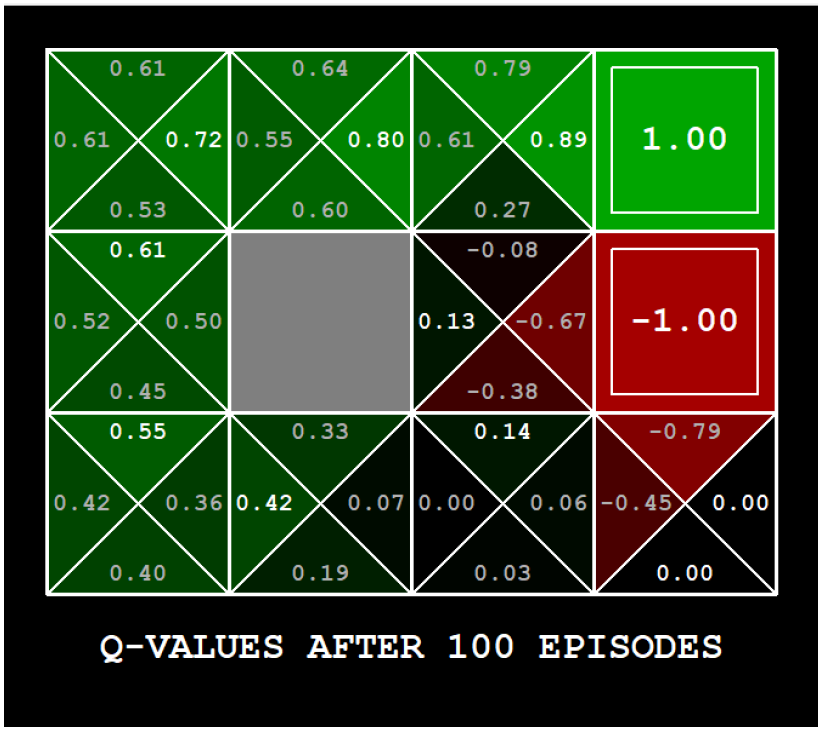
Task A – Experiment 1



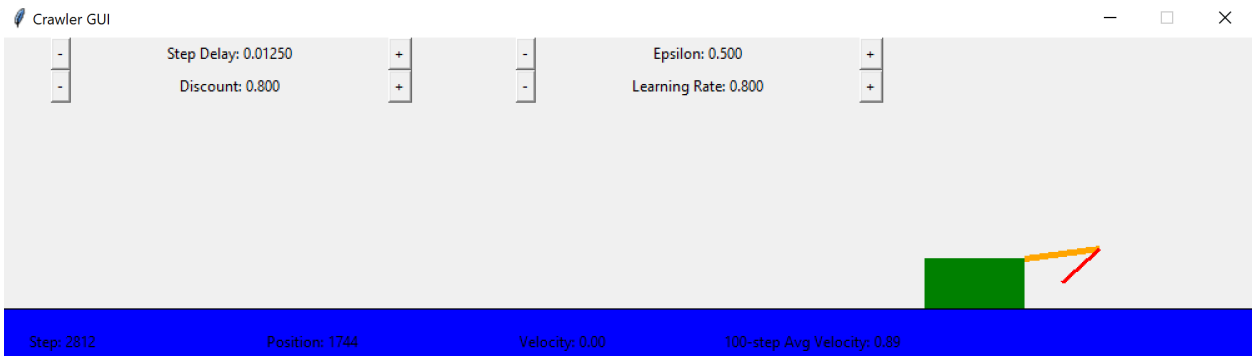
Task A – Experiment 2

```
In [2]: run autograder.py -q q3
Starting on 11-10 at 10:31:55

Question q3
===========

*** PASS: test_cases\q3\1-tinygrid.test
*** PASS: test_cases\q3\2-tinygrid-
noisy.test
*** PASS: test_cases\q3\3-bridge.test
*** PASS: test_cases\q3\4-
discountgrid.test

### Question q3: 5/5 ###


Finished at 10:31:55

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To
register your grades, make sure
to follow your instructor's guidelines to
receive credit on your project.
```

Task B – Experiment 1



Task B – Experiment 2



Task B – Experiment 3

```
In [1]: run autograder.py -q q4
Starting on 11-10 at 10:46:41

Question q4
===========

*** PASS: test_cases\q4\1-tinygrid.test
*** PASS: test_cases\q4\2-tinygrid-
noisy.test
*** PASS: test_cases\q4\3-bridge.test
*** PASS: test_cases\q4\4-
discountgrid.test

### Question q4: 2/2 ###


Finished at 10:46:42

Provisional grades
==================
Question q4: 2/2
------------------
Total: 2/2

Your grades are NOT yet registered.  To
register your grades, make sure
to follow your instructor's guidelines to
receive credit on your project.
```