CMSC 422: Assignment 5 – Reinforcement Learning – Spring 2019

The goal of this project is to give you some experience using basic Q learning of sequential actions. You will be provided with the file *sarsaBook.py* – a basic implementation of the Sarsa Algorithm (from p. 243 of Marsland's *ML* text) for learning how to traverse a simple "map" from location A to location F (Fig. 11.3, p. 234). It uses an ε-greedy policy for action selection.

After familiarizing yourself with *sarsaBook.py*, copy it into a new file *qlearn.py*, renaming the function *SARSA()* to be *qlearn()*. Use *qlearn.py* for the rest of this problem. Begin by modifying *qlearn.py* in the following three ways:

> Modify function *qlearn()* so that it implements the basic Q-Learning Algorithm (p. 242 of Marsland's *ML* text) rather than the Sarsa Algorithm. Basically, this involves modifying the line starting with Q[s,a] += mu * … near the end of the file.
>
> Noting that the greedy action selection steps used in the original *sarsaBook()* function, for example *a = np.argmax(Q[s,:])*, can select illegal actions at times, alter the <u>initialization</u> of Q values in *qlearn*() so that this will never happen (i.e., so that Q is consistent with the network structure and does not allow selection of illegal actions by a pure greedy policy, but is otherwise initialized the same way).
>
> Have *qlearn()* output the following information to a file *resultsQ.txt* during a run:
> - immediately after initializing Q, write out the arrays R, t and Q (neatly formatted, 1 row per line)
> - at the very end of a run, write out the final Q values (neatly formatted, one row per line)

Run the modified *qlearn()* function using the default parameters mu, gamma and epsilon that are embedded in the code, and then answer these questions based on the final Q values:

a. Draw a picture of locations (nodes) and all legal actions (links), similar to that of Fig. 11.3 in Marsland's *ML* text except now with directed links, where all of the learned Q values are written next to the corresponding links. For actions involving remaining at a location, write the Q values adjacent to the corresponding nodes.

b. If an agent uses the learned Q values shown in (a) and a purely greedy action selection process, what would its sequence of actions be when starting an episode in state A?

c. Was an optimal policy π learned? If so, why did the algorithm learn to follow the shortest path from A to the goal F, given that there are no explicit rewards built in for path shortness? If it did not learn an optimal policy, why not?

d. Are the learned Q values good estimates of the discounted cumulative reward that would actually be received by an agent starting in an arbitrary city and following a purely greedy policy until task completion? Justify your answer.

e. Given that the goal state F is supposed to be an absorbing state, what is wrong with the given transition matrix *t* ? (Don't change *t*, just explain what is inconsistent with being an absorbing state.) Why does that not cause any significant problem at run time?

Copy *qlearn()* into a new file named *qdecay()*. Make two changes in *qdecay()* as follows. First, add the statement Q = 0.95 * Q at the beginning of each step of learning (right after the line 'while s!=5:' at the start of the inner while-loop), allowing the Q values to decay somewhat during each step of learning.

Second, have *qdecay()* output the same information as *qlearn()* except now to a file *resultsDecay.txt* . Run *qdecay()* using the same parameter values.

f. Draw a picture of locations (nodes) and all legal actions (links), similar to that in (a), where all of the learned Q values obtained using decay are written next to the corresponding links and nodes.

g. How does Q value decay like this affect the final Q values relative to those obtained without decay?

What should I turn in?

The hardcopy and electronic submissions are due at different times. The hard copy portion is due at the start of class Tuesday April 30, the electronic submission is due earlier at 11:30 pm Monday April 29.

Hardcopy: Your answers to the questions (a)-(g).

Electronic submission: Turn in a single zip file that includes your code in *qlearn()* and *qdecay(),* the result files *resultsQ.txt* and *resultsDecay.txt*, and any other files you created to run the code (e.g., a run script). Use the Computer Science Department project submission server that is located at https://submit.cs.umd.edu to submit this zip file.