AINT351 MACHINE LEARNING 2015/16
P2.2: Q-LEARNING

Part of your AINT351 laboratory journal coursework

This set of laboratory practical exercises are part of the AINT351 laboratory journal coursework. After you have completed the exercises you must write a corresponding lab report and include it in the final lab journal as per the instructions provided in the AINT351 Coursework specification document available from the module DLE site.

Important note on practical

This should describe:

- What the lab tasks were
- How you tackled them (including code)
- Your results and findings.

This practical related to a classic reinforcement learning grid-word problems originally presented by McCallum (1996) and shown in Figure 1.

7	8	9	10	11
4		5		6
1		2 G		3

Figure 1: A grid-world with eleven states, the goal state, state 2, is indicated by the letter G. Originally presented by McCallum (1996).

1. Transition Function

States are numbered from 1 (bottom left) to 11 (top right) with the goal state being state 2, as shown in Figure 1. Actions are numbered 1 (north), 2 (east), 3 (south) and 4 (west). Write a deterministic transition function that takes a state and and action as parameters and returns the next state.

2. Starting State

Write a function that returns a random starting state. This state can be any of the states in Figure 1, bar the goal state.

3. Reward Function

Create a reward function that takes a state and an action and returns 10 if the state is 5 and the action is 3. In all other cases, it should return 0.

4. Q-function Table

Create a function initQ, that returns an initial Q-table, as an 11x4 matrix of random values between 0.01 and 0.1. Plot the initial Q-function table using a surface plot as shown in **Error! Reference source not found.**.

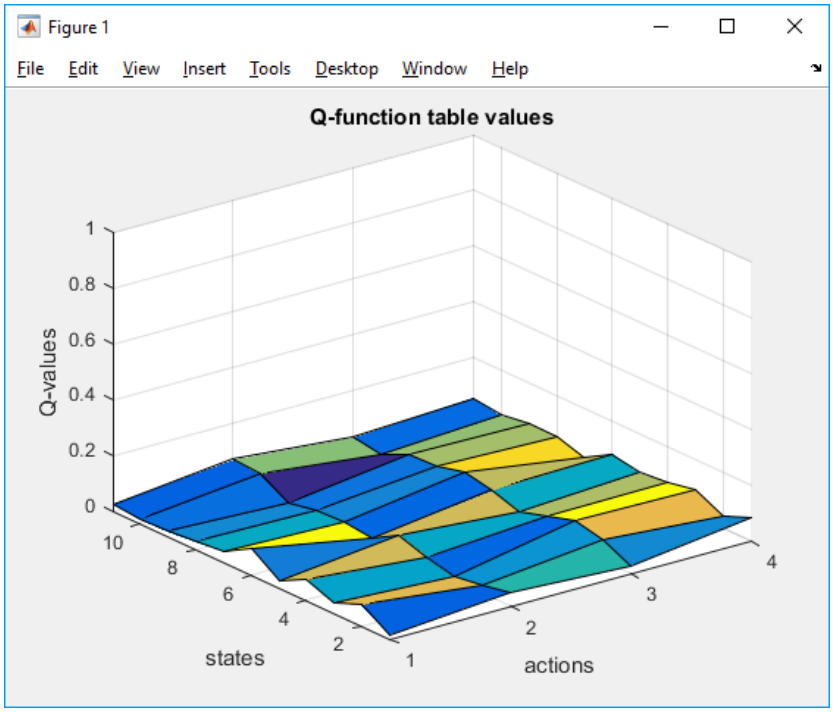

Figure 2: Q-function table initialized with random values between 0.0 and 0.1.

5. Action Selection

Implement an ε-greedy action selection function. Given a Q-table and a state as parameters, this function should return the action with the highest Q value 90% of the time. The remaining 10% of the time, the function should return a random action.

6. Update

Implement Q-learning update function that takes a Q-table, a state, an action, a resulting state and a reward value as parameters. The function should return the Q-table with one cell updated as per the Q-learning update rule given in line seven of the Q-learning algorithm in Figure 3. Use a temporal dicscount rate, γ, of 0.9 and a learning rate, α, of 0.2.

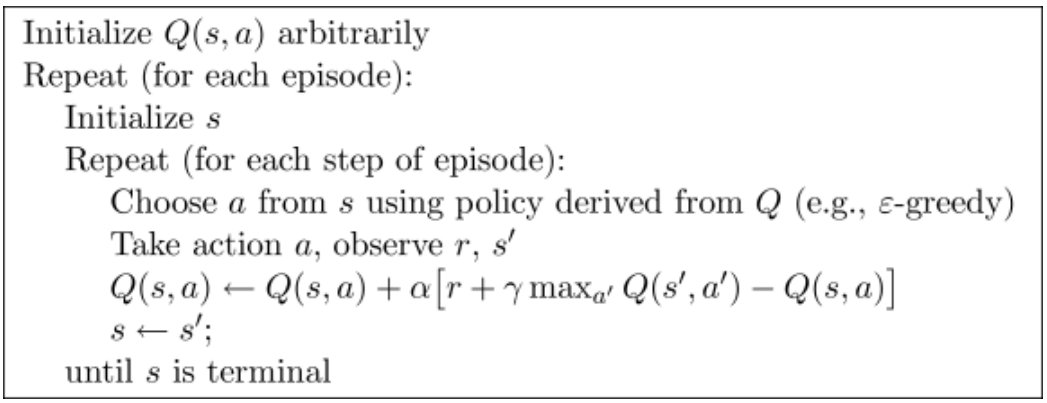

Figure 3: The Q-learning algorithm

7. Episode

Bring together the functions above in a single function that implements a Q-learning episode using the algorithm given in Figure 3. The state should be initialized to a random starting state at the start of each episode. The new function should also contain a loop that runs until the goal state is reached. The function should return the number of steps that were required to reach the goal state.

8. Trial

Write a trial function that runs 100 episodes. A trial should initialize the Q-table once and then use the same table for all episodes so that each episode can contribute to the overlall learing by updating it.

The trial function should return a vector containing the step counts for each episode. Plot this vector as a line plot as in Figure 4. Don't do the plotting as part of the trial function as this will be called repeatedly by the experiment function below.

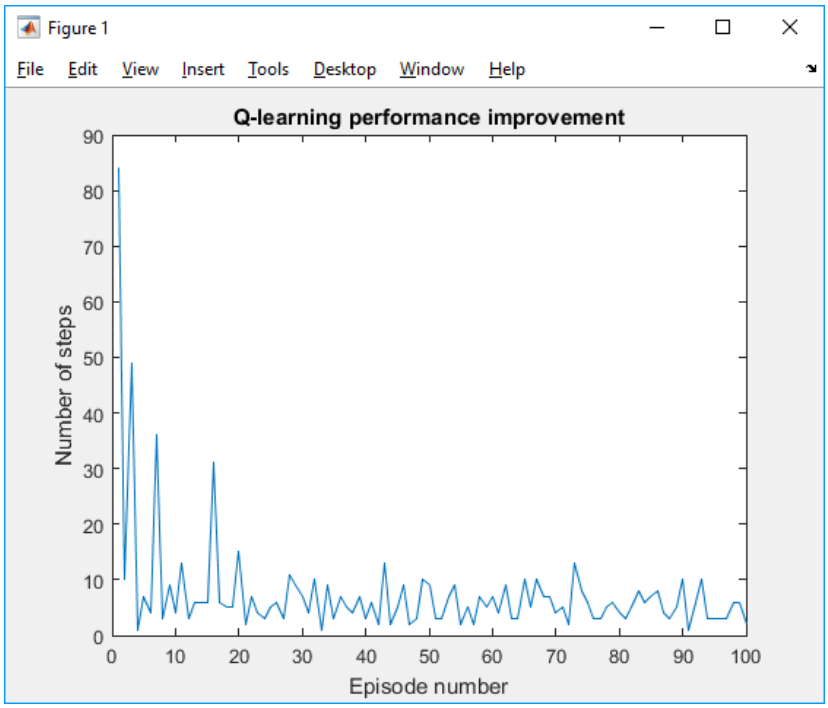

Figure 4: Trial data plot

9. Experiment

Write an experiment function that runs 500 trials and calculates the means and standard deviations in performance for each episode number. Plot the means and standard deviation using the 'shadedErrorBar' function[1] as shown in Figure 5.

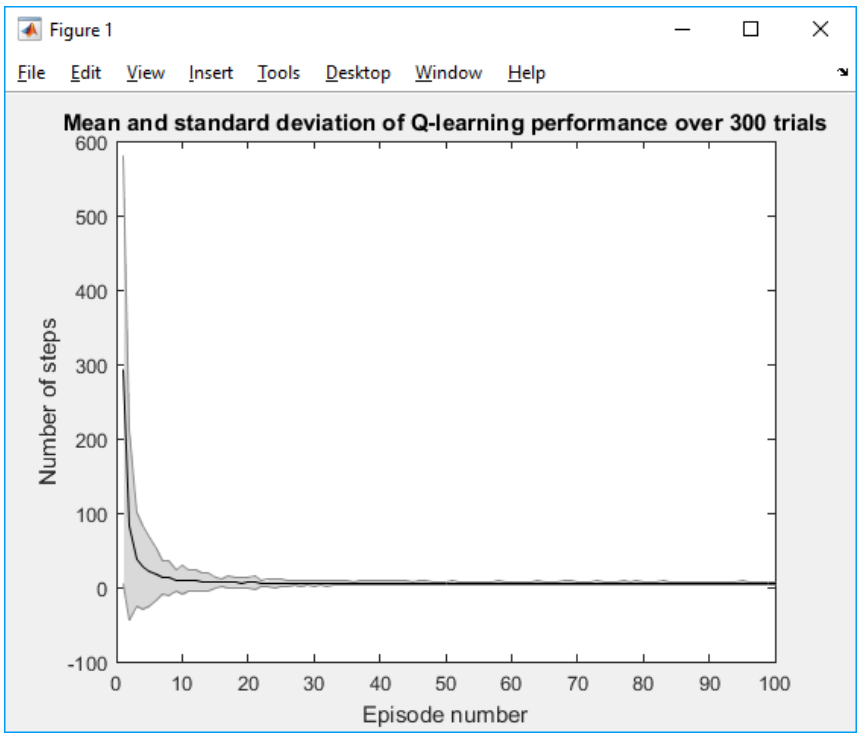

Figure 5: Experiment data plot

[1] The shadedErrorBar function is available at https://uk.mathworks.com/matlabcentral/fileexchange/26311-raacampbell-shadederrorbar. You must register for a MathWorks account using your Plymouth Univesity email in order to download it.