EE5904/ME5404 Part II

Project 2: Q-Learning for World Grid Navigation

Project Description and Requirement

Dr. Peter C. Y. Chen
Associate Professor
Department of Mechanical Engineering
National University of Singapore
Email: mpechenp@nus.edu.sg

Report due on April 24, 2020

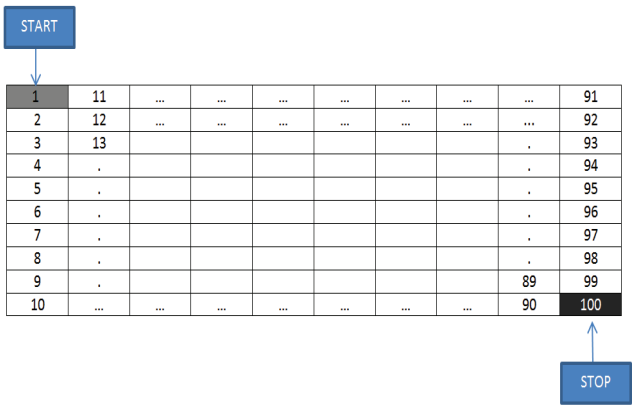
I. Objective

This project is designed for the student to demonstrate (through independent learning):

1. Competence in implementing the Q-learning algorithm, and
2. Understanding of the principles of, and implementation issues related to, the Q-learning algorithm.

II. Problem Statement

Suppose that a robot is to traverse on a 10×10 grid, with the start state being the top-left cell and the goal state being the bottom-right cell, as illustrated in Figure 1.



1	11	...	...	...	...	...	...	...	91
2	12	...	...	...	...	...	...	...	92
3	13							.	93
4	.							.	94
5	.							.	95
6	.							.	96
7	.							.	97
8	.							.	98
9	.							89	99
10	...	...	...	...	...	...	...	90	100

Fig. 1. Illustration of a 10×10 world grid with start state and goal state. The index of each cell follows the MATLAB column-wise convention.

The robot is to reach the goal state by maximizing the total reward of the trip. Note that the numbers (from 1 to 100) assigned to the individual cells represent the states; they do not represent the reward for the cells. At a state, the robot can take one of four actions (as shown in Figure 2) to move up ($a = 1$), right ($a = 2$), down ($a = 3$), or left ($a = 4$), into the corresponding adjacent state deterministically.

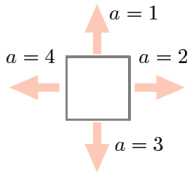


Fig. 2. Possible actions of the robot at a given state.

The learning process will consist of a series of trials. In a trial the robot starts at the initial state ($s = 1$) and makes transitions, according to the algorithm for Q-learning with ϵ-greedy exploration, until it reaches the goal state ($s = 100$), upon which the trial ends. The above process repeats until the values of the Q-function converge to the optimal values. An optimal policy can be then obtained.

III. Requirement

A. What to be done

There are two main tasks to be completed for this project.

Task 1: Write a MATLAB (M-file) program to implement the Q-learning algorithm, using the reward function as given in task1.mat and with the ϵ-greedy exploration algorithm by setting ϵ_k, α_k and γ as specified in Table I.

The file task1.mat is included in the zipfile that also contains this document. It can be directly loaded into MATLAB and contains the matrix variable reward (dimension: 100×4), in which each column corresponds to an action and each row to a state. For example, the reward for taking action $a = 3$ at state $s = 1$ to enter

state $s = 2$ is given by the $(1, 3)$ entry of reward, i.e., $\rho(1, 3, 2) = \mathsf{reward}(1, 3)$. Note that rewards can be negative.

TABLE I
PARAMETER VALUES AND PERFORMANCE OF Q-LEARNING

ϵ_k, α_k	No. of goal-reached runs		Execution time (sec.)	
	$\gamma = 0.5$	$\gamma = 0.9$	$\gamma = 0.5$	$\gamma = 0.9$
$\frac{1}{k}$	?	?	?	?
$\frac{100}{100+k}$	?	?	?	?
$\frac{1+log(k)}{k}$	?	?	?	?
$\frac{1+5log(k)}{k}$	?	?	?	?

In this task, ϵ_k and α_k are set to the same value. You are required to run your program 10 times (i.e., 10 runs) for each set of parameter values and record the number of times the goal state is reached. (*Note: It is possible that some of the runs may not yield an optimal policy that results in the robot reaching the goal state; these runs are not to be counted.*) The maximum number of trials in each run is set to 3000. The average program execution time of the "goal-reaching" runs is to be calculated and entered into the table (as indicated by "?"). The final output of your program should be an optimal policy (if the goal state is reached in any of the 10 runs), and the reward associated with this optimal policy. The optimal policy is to be expressed as a column vector containing the state transitions, and also illustrated (in your report) on a 10×10 grid with arrows marking the trajectory of the robot moving from state $s = 1$ to state $s = 100$.

Task 2: Write a MATLAB (M-file) program to implement Q-learning using your own values of the relevant parameters. Assume that the grid size is 10×10 and implement your program in a MATLAB M-file. This M-file will be used to find the optimal policy using a reward function not provided to the students, as part of the assessment scheme discussed in Section V.

B. What to submit

1. A report (in a PDF file) describing the implementation and the results. The name of the PDF file must be in the format:

   StudentNumber_RL.pdf

   This report should be no more than ten pages. The report must contain a cover page showing: (i) **student's name**, (ii) **student number**, (iii) **student's email address**, (iv) **name of module**, and (v) **project title**.
2. The M-file programs as specified in the description of **Task 1** and **Task 2** in Section III-A above.

C. How to submit

Only softcopy of the report (in PDF) and the MATLAB M-file programs are to be submitted. Please put the report and the M-file programs in a folder. **Use your student number as the folder name**. Generate a non-password-protected zipfile of this folder and upload this zipfile to LumiNUS at:

Module: EE5904/ME5404 Neural Networks [1920] 2019/2020 Semester 2
Folder: Part II RL project report submission

Note: Make sure to upload your report into the correct folder as specified above.

IV. DEMO SESSION

A demo session, to be conducted by the teaching assistant, on the use of MATLAB for implementing the Q-learning algorithm will be held during one of the lectures. Please check the schedule in the lecture slides for the specific date. Students are strongly advised to attend this session.

V. ASSESSMENT

The project will be assessed based on the following criteria:

1. *Comments (with supporting argument) on the results obtained in Task 1*
2. *Presentation*. This includes good report style, clarity, and conciseness.
3. *Performance of your M-file program for* **Task 2** *(as described in Section III-A) in finding an optimal policy based on the reward function specified in a file* qeval.mat. Your M-file program must be workable in MATLAB under the Windows environment. When qeval.mat is loaded into MATLAB, the MATLAB workspace will have a variable named qevalreward whose dimension is 100×4, with each column corresponding to an action and each row to a state – similar to the variable reward described above in **Task 1**. Your M-file program must be able to process and generate as fast as possible a column vector named qevalstates as output, whose n^{th} element is the state visited in the n^{th} transition. Also output a 10×10 grid showing the trajectory of the robot, along with the total reward. You can assume that the variable qevalreward is available in the MATLAB workspace when your M-file is run during assessment. When writing your M-file program, you can test its execution on your own by making up a qeval.mat file containing dummy sample values.