Q-Learning for World Grid Navigation

EE5904/ME5404 Part II: Project 2

Report due on 26 April 2019

Yáng YuánRuì

yang_yuanrui@u.nus.edu

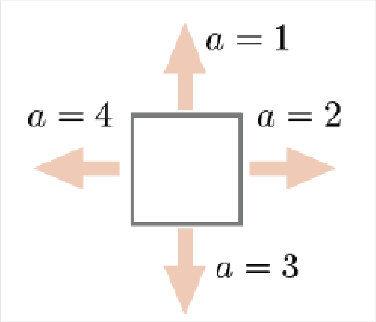
11 April 2019

Outline

- Project Description
- Recap
- Project Implementation
- Important Notes

Project Description-Task

- The robot is to reach the goal state with maximum total reward of the trip



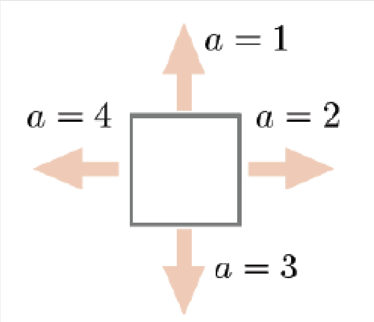
START

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

STOP

Illustration of a 10×10 world grid with start state and goal state. Index of each cell follows the Matlab column-wise convention for ease of programming

Project Description: State Transition



- At a state, the robot can take 1 of the 4 actions
- The state transition is **deterministic**
- Assuming the state transition simulation is given by the **deterministic** state transition model
- You can actually use dynamic programming to find the optimal policy since the “model” is given
- In this project, you are only required to implement ***Q*-Learning**
- Some of the actions are not allowed, for the states moving out of the boundary

Project Description: Reward Function

- Reward is given as a matrix “task1.mat” (known in Task 1)
- Reward Matrix:
  - Dimension: 100×4
  - Each column represents an action (4 actions)
  - Each row represents a state (100 states)
- Prohibited transitions are marked by a reward of **–1**

Recap

- Total Reward for an agent continuing its transition:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

  - R_t determines present value of future rewards
  - Rewards received k steps in the future is discounted by factor γ^{k-1}
  - Small γ forces agent to focus more on intermediate rewards from next few steps
  - Large γ forces agent to take into account future rewards more strongly (agent becomes more farsighted)

Recap

- 'Worth' actions at different states

$$Q^{\pi} : S \times A \rightarrow \boxed{\mathcal{R}}$$

- $Q^{\pi}(s, a) = E^{\pi}[R_t | s_t = s] \dashrightarrow R_t | s_t = s$

  Deterministic Transition

  - Expected return from taking action a at sate s at time step t by following action $\boldsymbol{\pi}$

- Optimal policy is one that maximizes values of Q-functions overall all possible (s, a)

Recap: Optimal Policy

$$Q^\pi(s,a) \quad = \quad E^\pi\left[r_{t+1}\right] + E^\pi\left[\gamma \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+2} \middle| \, s_t = s\right]$$

Slides 164-166

Values of Q-function are optimal if they are greater or equal to that of all other policies for all (s,a) pairs, i.e.,

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

Greedy policy

At each s, select a that yields the largest value for the Q-function. When multiple choices are available, such a can be picked randomly

Optimal policy: $\pi^*(s) \in \arg\max_a Q^*(s,a)$

Dynamic programming when state-transition model is given.

Recap: Model-Free Value Iteration

When state transition model is **unknown**, the Q-function can be estimated via iterative update rule by using the reward received from observed state transition

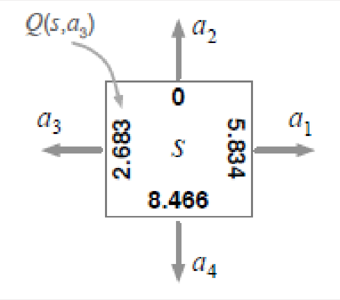
$$
\begin{aligned}
& Q_{k+1}(s_k, a_k) \\
= \quad & Q_k(s_k, a_k) + \alpha_k \left(\underbrace{\overset{\text{Reward of action } a \text{ at state } s}{\boxed{r_{k+1}}} + \gamma \max_{a'} Q_k\left(s_{k+1}, a'\right)}_{\text{Estimate of } Q^*(s_k, a_k)} - Q_k(s_k, a_k) \right)
\end{aligned}
$$

Exploitation: use **greedy** policy to select currently known best action

$$a_{k+1} = \max_{a'} Q_k(s_{k+1}, a')$$

Exploration: **Try** action other than current known best action

$$a_{k+1} \neq \max_{a'} Q_k(s_{k+1}, a')$$



Exploitation: Take a_4

Exploration: Take a_1, a_2, a_3

Recap: ϵ-greedy exploration

Input: Discount factor γ; exploration probability ϵ_k; learning rate α_k

Initialization

- Initialize Q-function, e.g., $Q_0 \leftarrow 0$
- Determine the initial state s_0
- For time step k, select action a_k according to:

Select Action

Exploitation

Exploration

$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1 - \epsilon_k \\ \text{an action uniformly randomly selected from all other actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

Apply Action

- Apply action a_k, receive reward r_{k+1}, then observe next state s_{k+1}
- Update Q-function with:

Update Q-value

$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left(r_{k+1} + \gamma \max_{a'} Q_k\left(s_{k+1}, a'\right) - Q_k(s_k, a_k) \right)$$

- Set $k = k + 1$ and repeat for-loop for the next time step

Implementation: Q-Learning

- Initialize
- For each trial
  - For each move
    - Select action
    - Apply action
    - Update Q-Value
- Extract Optimal policy

Implementation: Parameter Setup

Input: Discount factor γ; exploration probability ϵ_k; learning rate α_k

- Initialize Q-function, e.g., $Q_0 \leftarrow 0$
- Determine the initial state s_0

TABLE I

PARAMETER VALUES AND PERFORMANCE OF Q-LEARNING

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

$\boldsymbol{\epsilon_k = \alpha_k}$

k is time step

Implementation: For each move

- Select action
- Apply action
- Update Q-Value

- For time step k, select action a_k according to:

$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1 - \epsilon_k \\ \text{an action uniformly randomly} \\ \text{selected from all other actions} \\ \text{available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

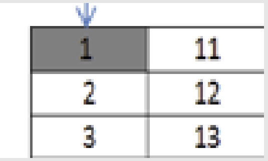
- Apply action a_k, receive reward r_{k+1}, then observe next state s_{k+1}
- Update Q-function with:

$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left(r_{k+1} + \gamma \max_{a'} Q_k\left(s_{k+1}, a'\right) - Q_k(s_k, a_k) \right)$$

- Set $k = k + 1$ and repeat for-loop for the next time step

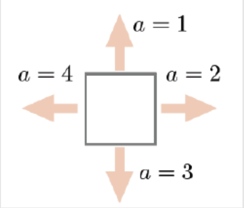
Implementation: For each move

- For the k^{th} movement, the machine is at state i
- Choose the next movement based on ϵ, ($e.g.$, $\epsilon = \frac{1}{k}$)
- Assume the current optimal movement is to the left ($i.e.$ $a_k = 4$)
  - Exploitation action is $a_k = 4$, with probability of $1 - \epsilon = 1 - \frac{1}{\mathrm{k}}$
  - Exploration action is $a_k = 1{,}2{,}3$, each with probability of $\frac{\epsilon}{3} = \frac{1}{3k}$
    - What if i is at boundary?
    - Exploration action are uniformly selected from those possible explorative actions



- Assume $a_k = 2$ is taken (i.e., move to the right)
  - $Q(i,2) = Q(i,2) + \alpha_k(reward(i,2) + \gamma \max\big(Q(i+10,:)\big) - Q(i,2))$

given from task1.mat



Implementation: Termination Condition

In theory:

- **Trial termination condition:**
  - In each trial, the robot starts at initial state ($s = 1$)
  - It makes a series of transitions according to the algorithm for Q-learning with ϵ-greedy exploration
  - A trial ends when the robot reaches the goal state ($s = 100$)

- **Number of trials:**
  - Repeat the process until the values of the Q-function converges to the optimal values

Implementation: Termination Condition

In this project:

- **Trial termination condition:**
  - The robot reaches the goal state ($s = 100$), *or*
  - $\alpha_k < 0.005$ (recommended)
    - You may also try other threshold

- **Number of trials:**
  - Q-function converged to the optimal values, *or*
  - Number of trials ≥ 3000 (Try other options also)

Implementation: Overview

- For each trial
  - For each move
    - Select action (perform exploitation or exploration)
    - Apply action
    - Update Q-value

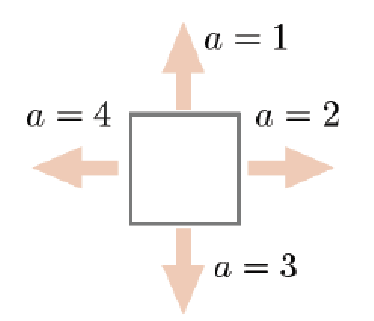
$$Q_{next}(s,a) = Q_{crt}(s,a) + \alpha_k\left(reward(s,a) + \gamma \max\left(Q_{crt}(s',:)\right) - Q_{crt}(s,a)\right)$$

    - Check trial termination condition
  - Check Q-value convergence / program termination condition
- Use the Q_{final} to extract the optimal path with greedy policy:

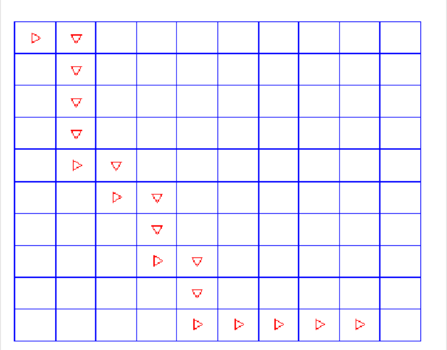
$$\pi^*(s) \in \arg\max_a Q^*(s,a)$$

Implementation: Plot

- Show arrow
  - `plot(x, y, '^')`%, action 1
  - `plot(x, y, '>')`%, action 2
  - `plot(x, y, 'v')`%, action 3
  - `plot(x, y, '<')`%, action 4



- Show the grid world
  - plot lines to show the grid
- Subplot
  - easier viewing



For illustration only

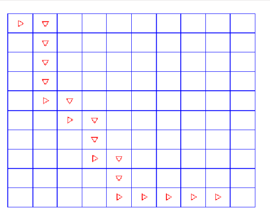
Important Notes: Task 1

TABLE I

PARAMETER VALUES AND PERFORMANCE OF Q-LEARNING

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

1. Matlab code are executable which generates reported results with the provided task1.mat
2. Complete Table I
3. Plot a 10×10 grid showing the trajectory and the total reward
4. Provide necessary discussion based on your experimental results



Important Notes: Task 2

- Choose a learning rate and discount rate wisely so that your robot can deal with the unknown reward.
- The code will be used to find the optimal policy using a reward function *not* provided to the students.
- "qeval.mat" will be used (as a replacement for the reward you have from task1.mat) to evaluation your RL program.

Important Notes: Task 2

- You can assume that the unknown reward "`qevalreward`" are loaded in the workspace
- The state transition model, initial state, goal state, are the same as those in Task 1, but the reward function is different.
- The output of your program should be a column vector named "`qevalstates`" to store the trajectory
- Also plot a 10×10 grid showing the trajectory and the total reward

Important Notes: Task 1 & 2

- The program evaluation will be based on
  - Policy
  - Execution time
  - *Executable*!
- You can create your own reward matrix to test the effectiveness of your code
- The marking will be based on the report and the program code. Explain your choice of parameters clearly in your report.

Important Notes

- Name your RL main script (for Task 2) as "RL_main" for testing unknown reward.

- Please play around with the discount factor, the learning rate and the exploration probability.

- Use your student number as the folder name. Generate a non-password-protected zipfile of this folder and upload this zipfile to the IVLE.

Report due on 26 April 2019

Thank you!