EE5904/ME5404 PART II
PROJECT 2:
Q-LEARNING FOR WORLD GRID NAVIGATION

Hanyu Bai

e0816201@u.nus.edu

PROJECT DESCRIPTION

Objectives

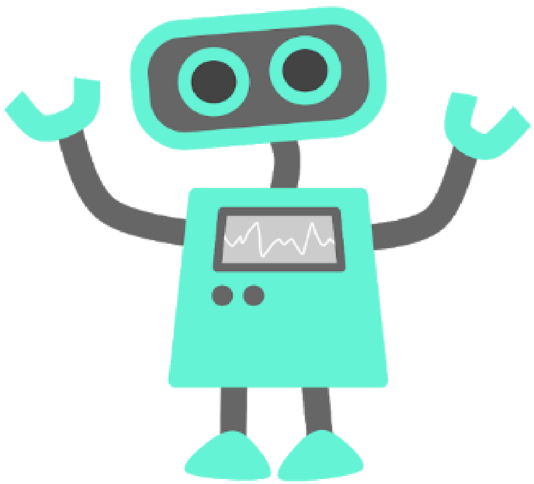
1. Competence in implementing the Q-learning algorithm
2. Understanding of the principles of, and implementation issues related to, the Q-learning algorithm.

Contents

1. Tasks
2. State transition
3. Reward function
4. Learning

TASK

Using Q-learning with ϵ-greedy exploration. The robot is to move from the initial state (s = 1) to the goal state (s = 100) with the maximum total reward of the trip.



START

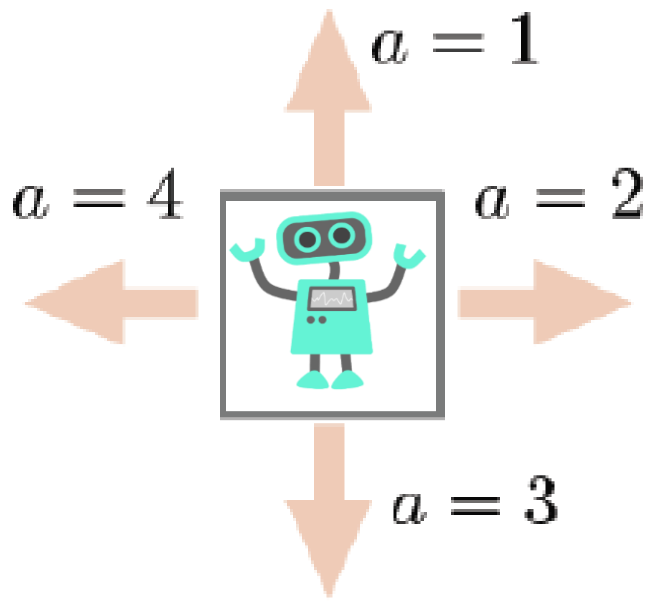
1	11	...	...	...	...	...	...	...	91
2	12	...	...	...	...	...	...	...	92
3	13							.	93
4	.							.	94
5	.							.	95
6	.							.	96
7	.							.	97
8	.							.	98
9	.							89	99
10	...	...	...	...	...	...	...	90	100

STOP

STATE TRANSITION

At a state, the robot can take one of four actions

Use dynamic programming methods to find the optimal policy



REWARD FUNCTION

Files

- Task 1 → "reward" in "task1.mat"
- Task 2 → "qevalreward" in "qeval.mat"

Reward Matrix:

- Column → Action
- Row → State
- 100 x 4

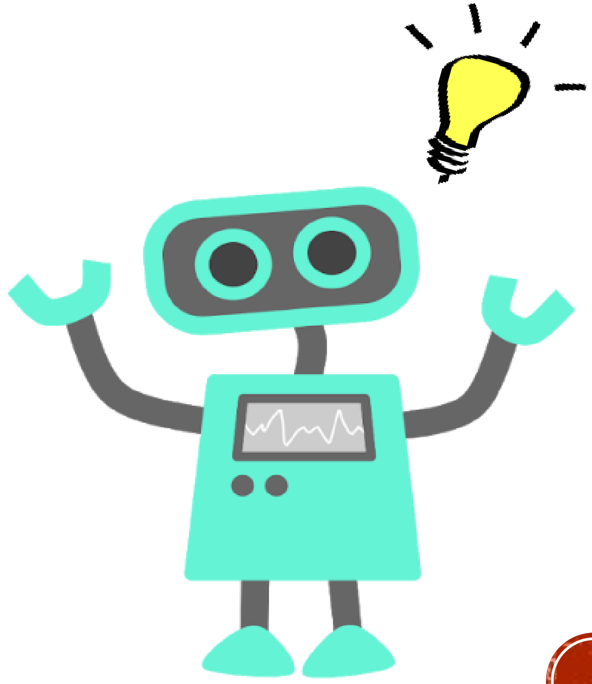
$$\begin{bmatrix} r_{1\,1} & \cdots & r_{1\,4} \\ \vdots & \ddots & \vdots \\ r_{100\,1} & \cdots & r_{100\,4} \end{bmatrix}$$

Actions

States

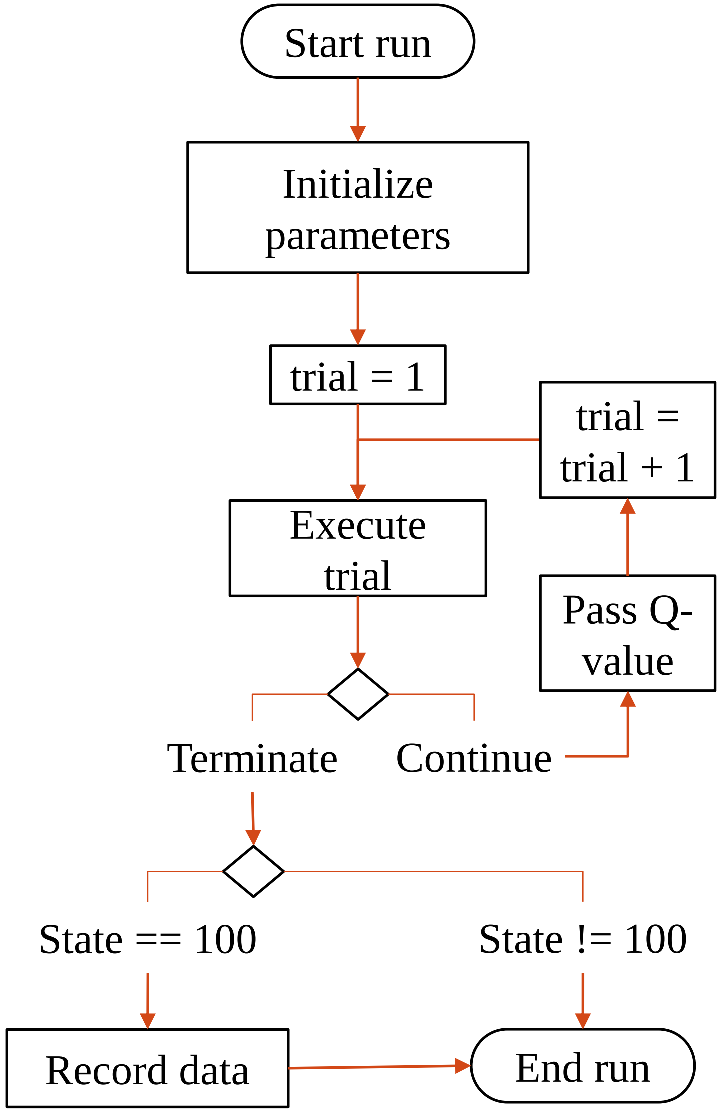
LEARNING

- The robot learns in 1 run, and one run consists of the **N trials**
- Each run starts with a set of initial values of the Q-function (100 x 4 matrix)
- Each trial starts when the robot moves from state 1, and ends when the robot reaches state 100
- The Q values are passed to the next trial
- Each run ends when the Q values **converge to the optimal values**

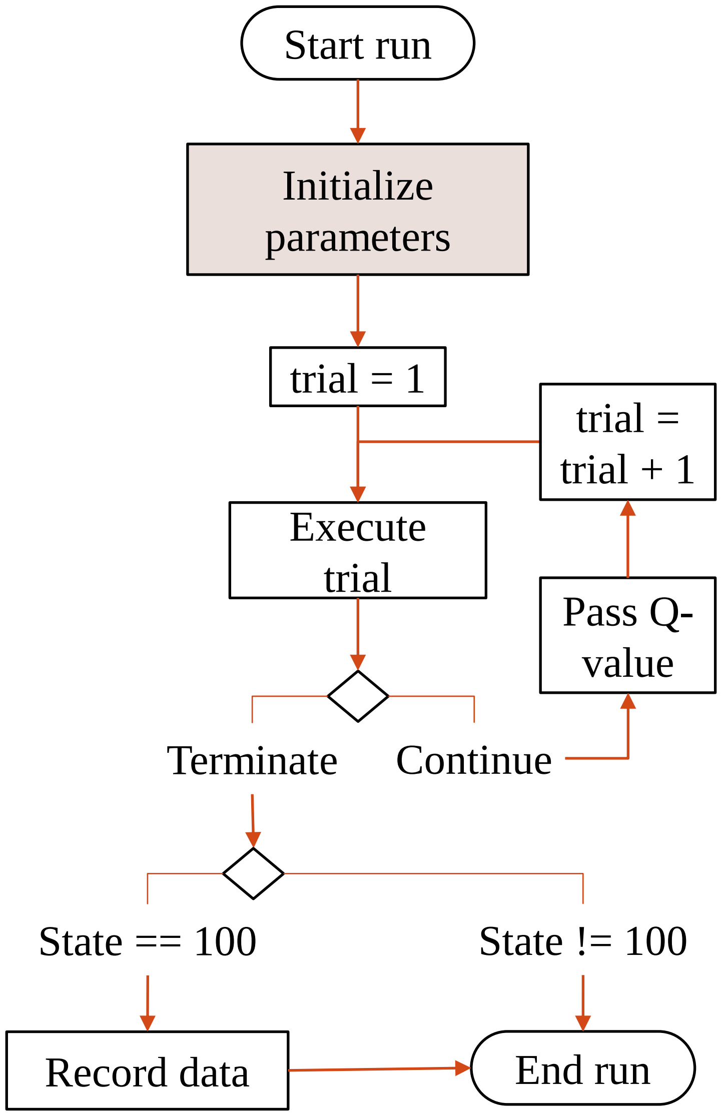


IMPLEMENTATION

IMPLEMENTATION



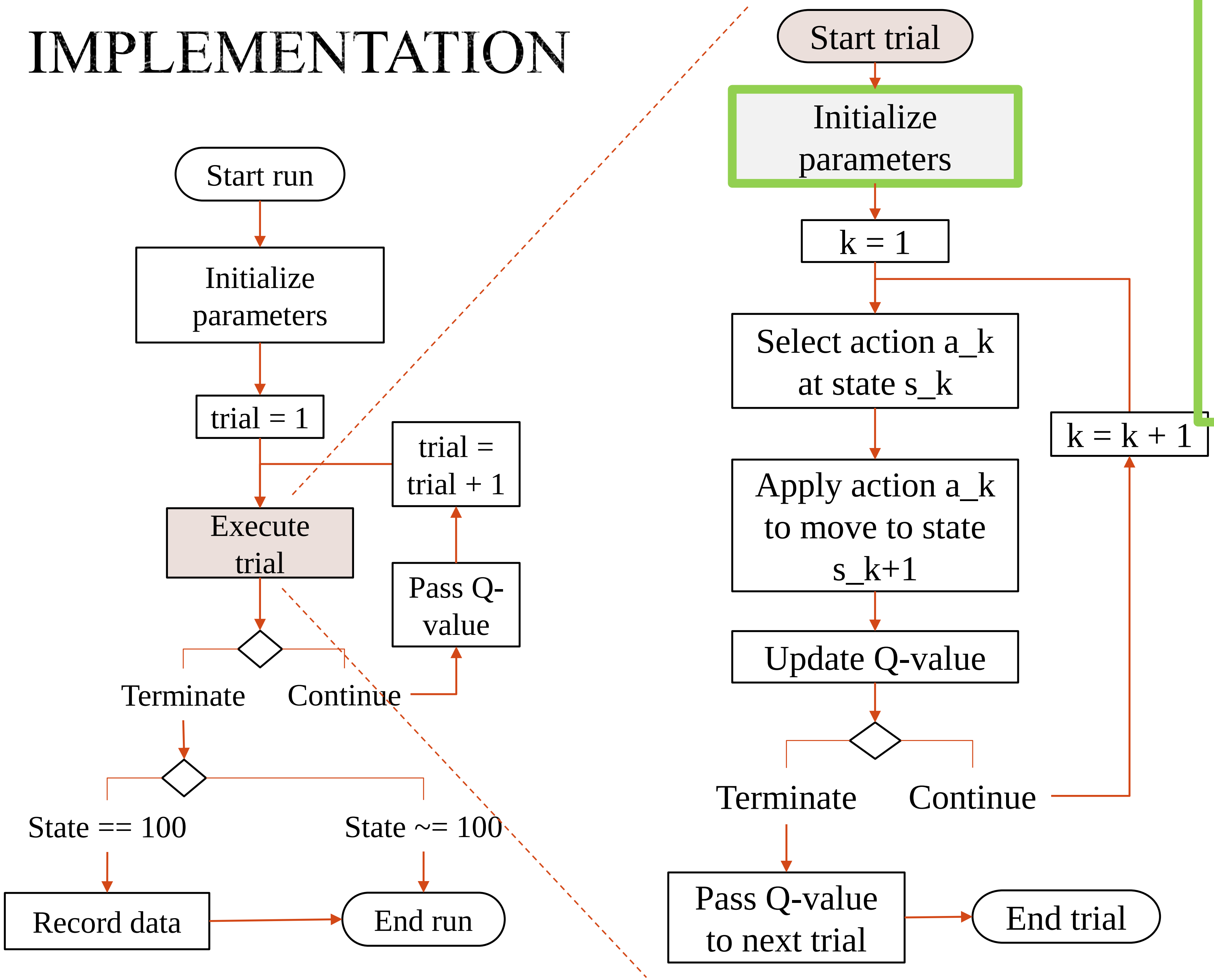
IMPLEMENTATION



Parameters:

- Initial Q-function $Q_1 \leftarrow 0$ >> Optional
- Trial = 1
- threshold of error of Q-values between trials

IMPLEMENTATION



Parameters:

- Discount factor γ
- Exploration probability ϵ_k
- Initial Q-function Q_1

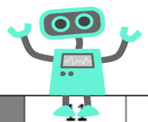
  From previous trial, if any
- Learning rate $\alpha_k = \epsilon_k$
- Initial state $s_1 = 1$
- Time step $k = 1$

IMPLEMENTATION

Example:

For $k = 3$,

- $\epsilon = 1 - \frac{1}{k}$
- $s_3 = 11$



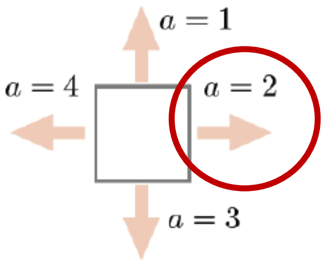
1	11	...
2	12	...
3	13	
4	.	

Recall that:

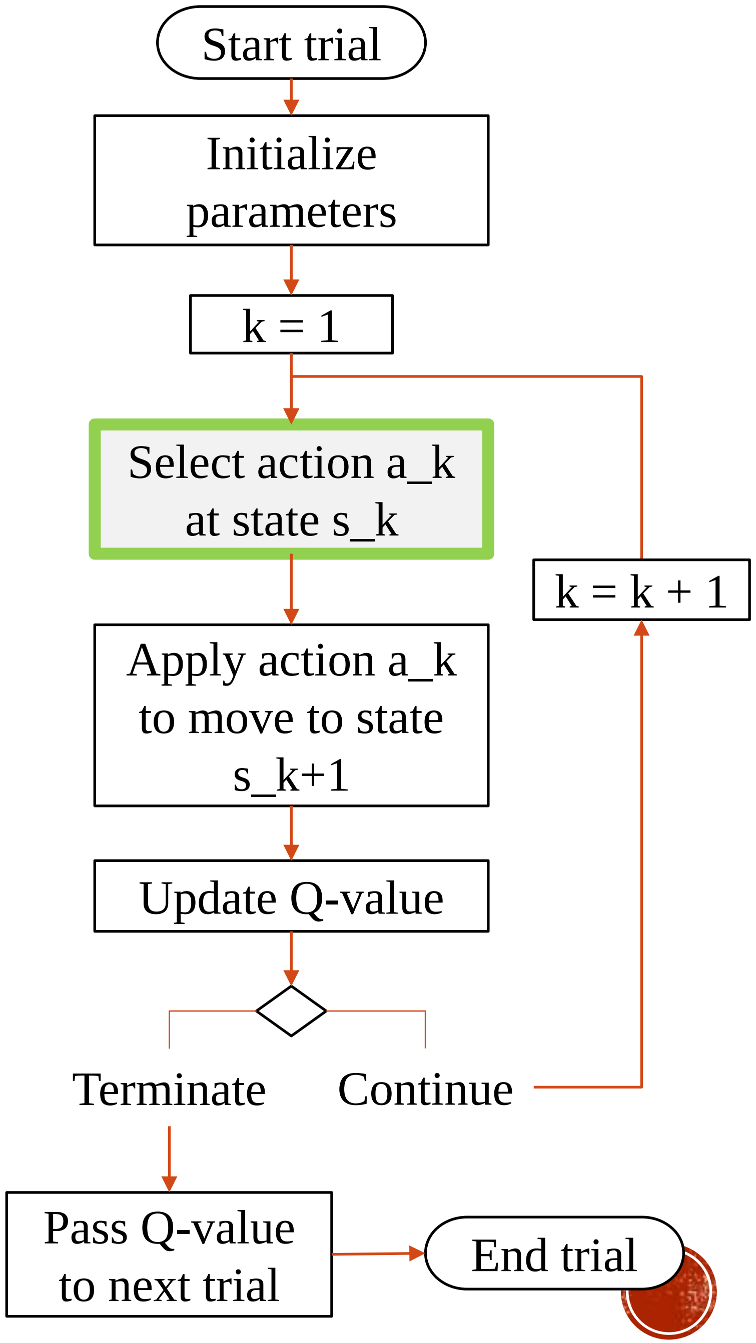
$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1-\epsilon_k \\ \text{an action uniformly randomly selected from all other actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

Current best action is 2

- <u>Exploitation</u>: $a_3 = 2$ each has $1 - \epsilon = \frac{1}{3}$ probability to be selected
- <u>Exploration</u>: $a_3 = 3, 4$ each has $\epsilon = \frac{2}{(2)3}$ probability to be selected



*** $a_3 = 1$ **cannot be** selected due to the **boundary**

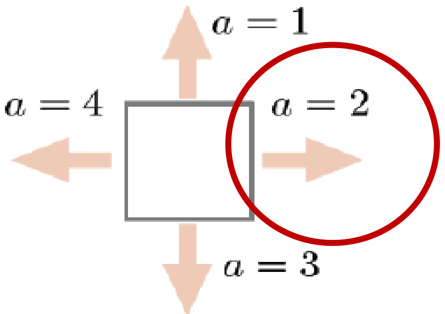


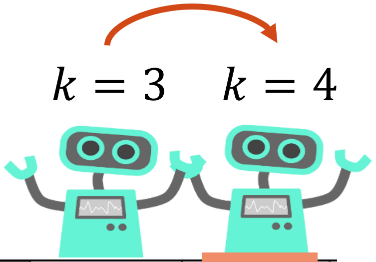
IMPLEMENTATION

Example (continue):

For $k = 3$,

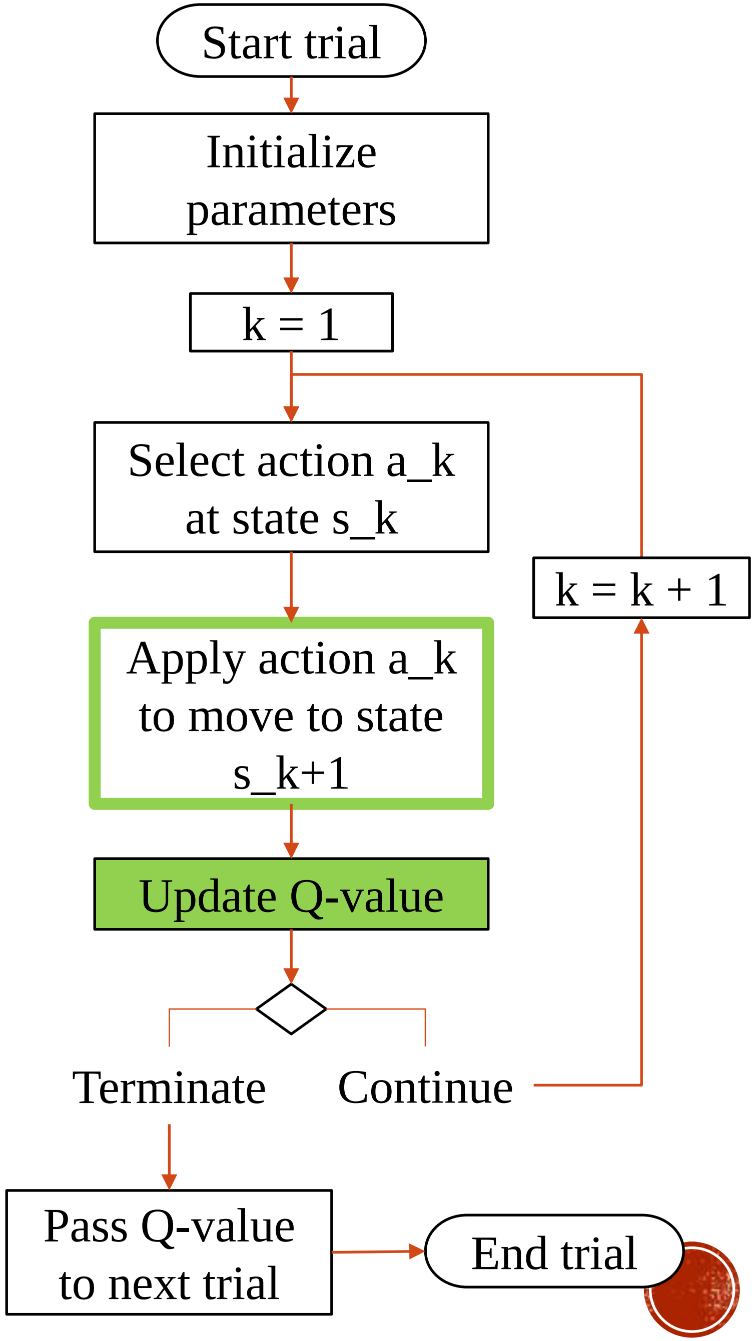
- $s_3 = 11$
- Selected action **is $\mathbf{a_3 = 2}$**
- $s_4 = 21$





1	11	21
2	12	...
3	13	
4	.	

- Receive reward $r_{11\ 2}$
- $Q_4(11,2) = Q_3(11,2) + \alpha_3(\boldsymbol{reward}(11,2) + \gamma * max\big(Q_3(21,:)\big) - Q_3(11,2))$



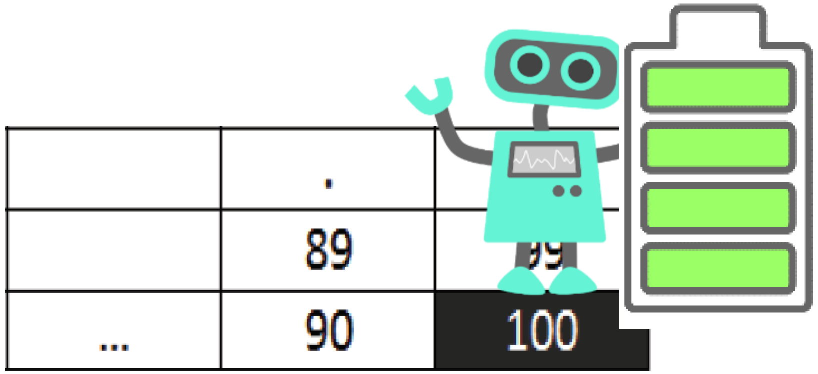
IMPLEMENTATION

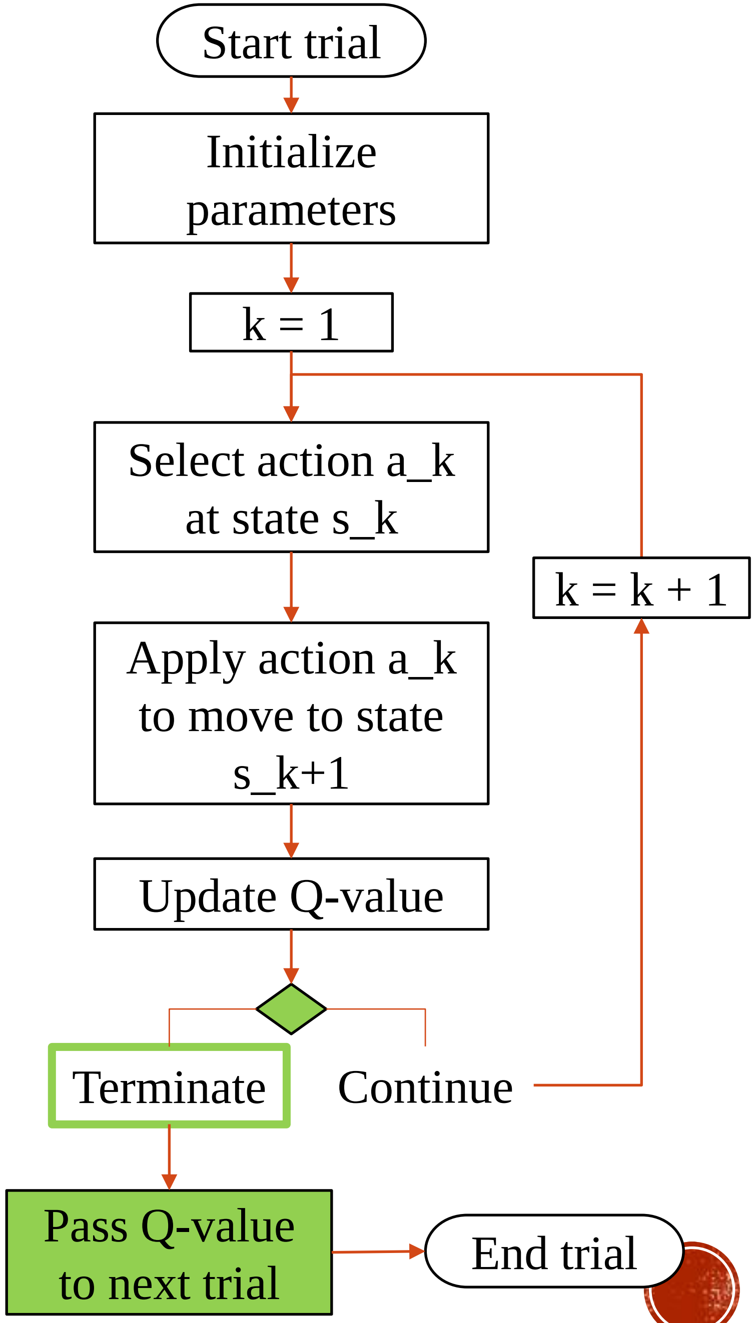
Termination condition for <u>**each trial**</u>:

- Robot reaches goal state $s_k = 100$ >> **Ideal Case**
- $\alpha_k < 0.005$ >> **Optional**
- Maximum number of time step $k_{\max}$ is reached

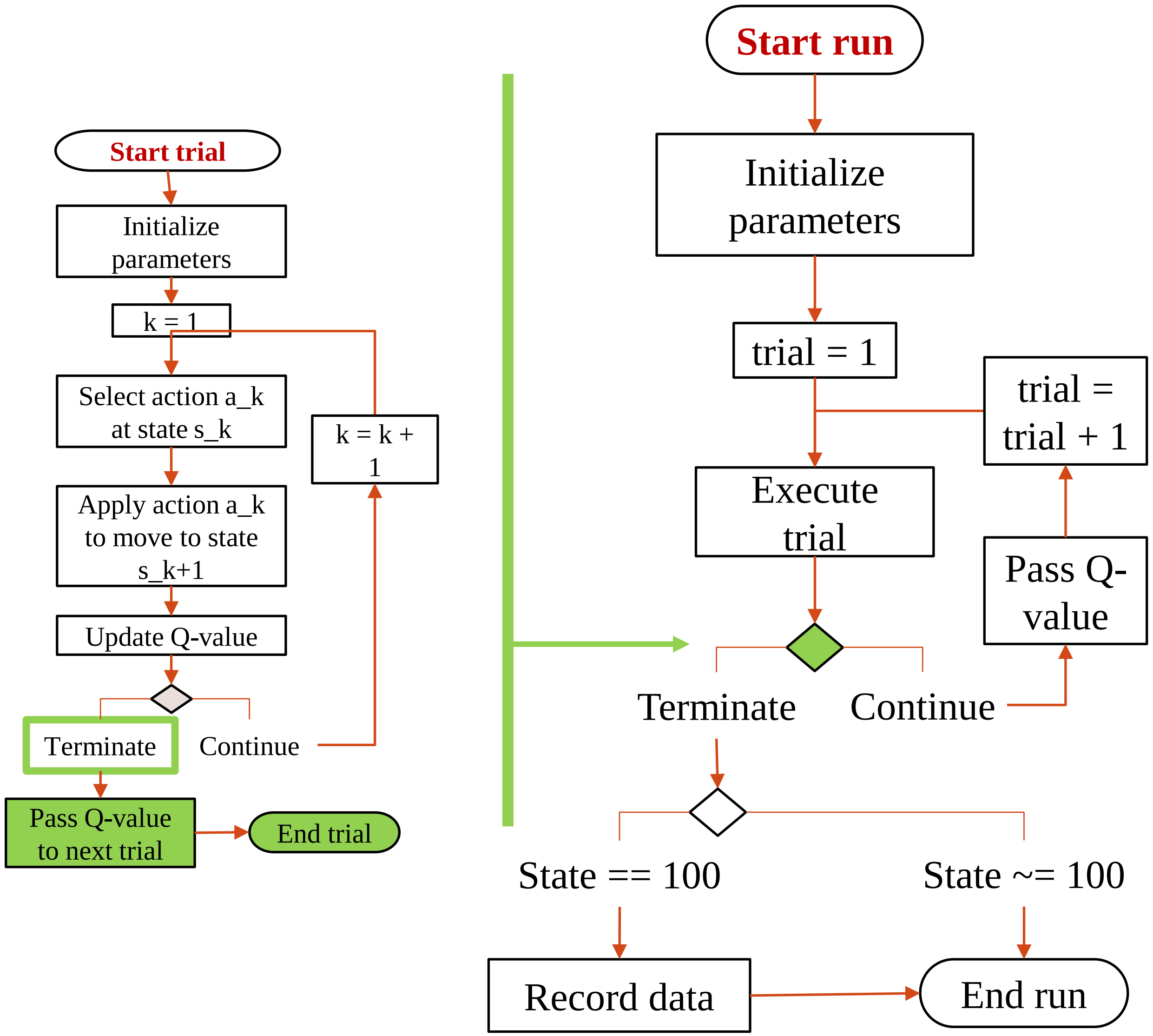
Continuation condition for each trial:

- Otherwise





IMPLEMENTATION



Initial Q-values of **next trial** is the optimal policy of this trial.

Termination condition for **each run**:

- Q-function converged to the optimal values **>> Ideal Case**
- Maximum number of trials $trial_{max}$ is reached

Continuation condition for each trial:

- Otherwise

TASK I

WHAT TO DO?

1. Implement Q-learning algorithm in MATLAB
2. Reward function is in task1.mat
3. Discount factor γ and exploration probability ϵ_k are given in Table 1
4. For each set of parameter values,
   - Run 10 runs ($trial_{max} = 3000$ each)
5. Complete report

TABLE I

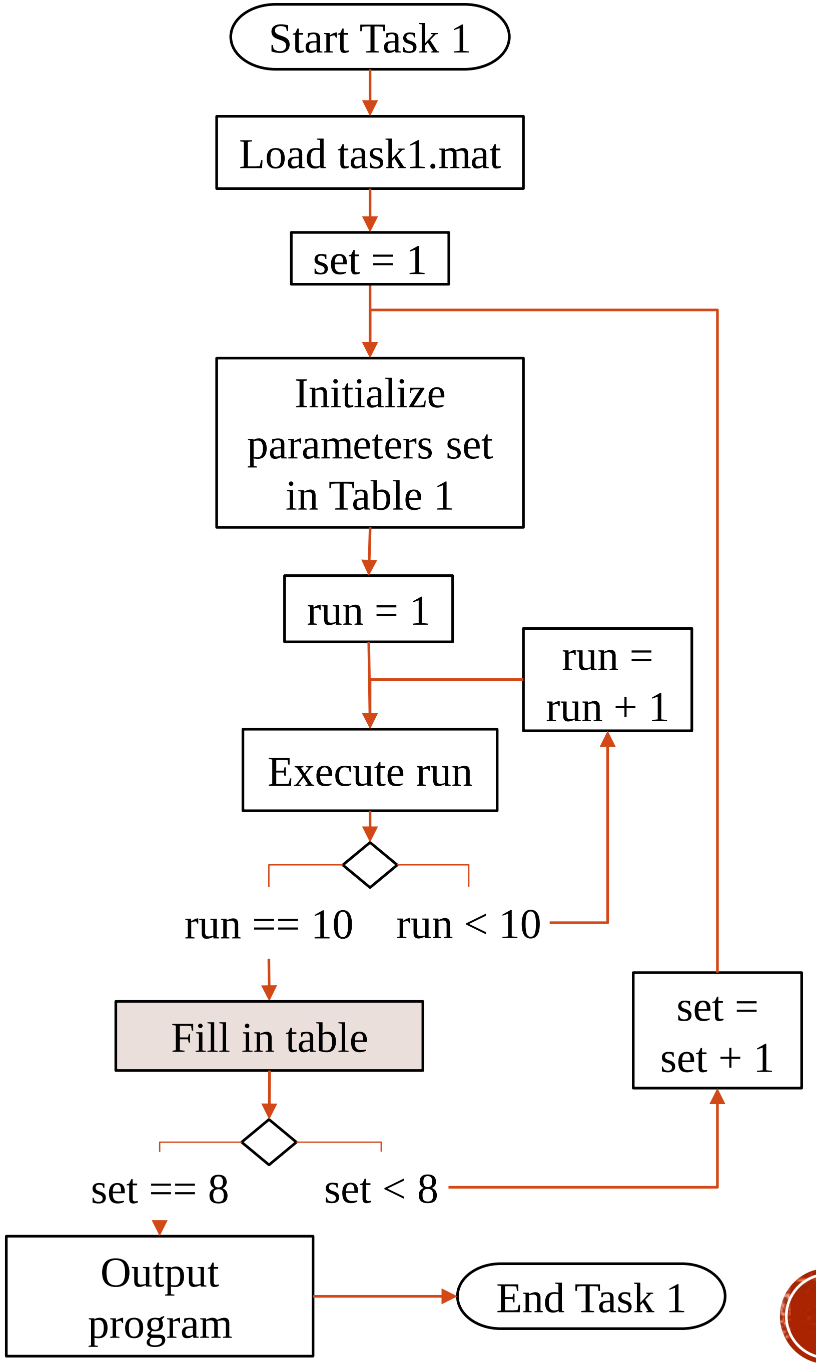
PARAMETER VALUES AND PERFORMANCE OF Q-LEARNING

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

For each parameter set in Table 1:

Number of goal reaching runs:
Out of 10 runs, count the runs that the robot ends at state 100.

Execution time:
Average the recorded execution time for those goal reaching runs.

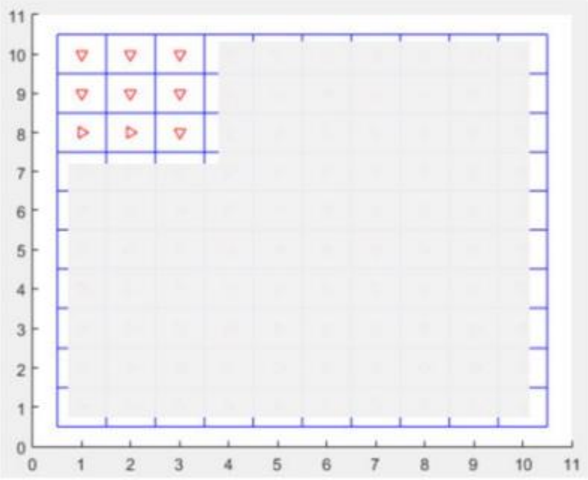


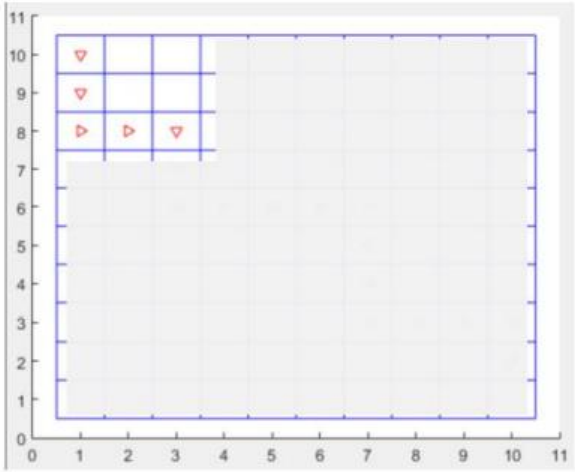
OUTPUTS

1. As a column vector, with the position in the column corresponding to a state, and the entry for that position representing the action selected by the optimal policy at that state.

2. As a 10×10 grid diagram **with arrows** indicating the action selected by your optimal policy at each state.

3. As a 10×10 grid diagram showing **the (optimal) path** taken by the robot as it moves from the initial state to the goal state according to your optimal policy, plus the reward associated with this optimal path.





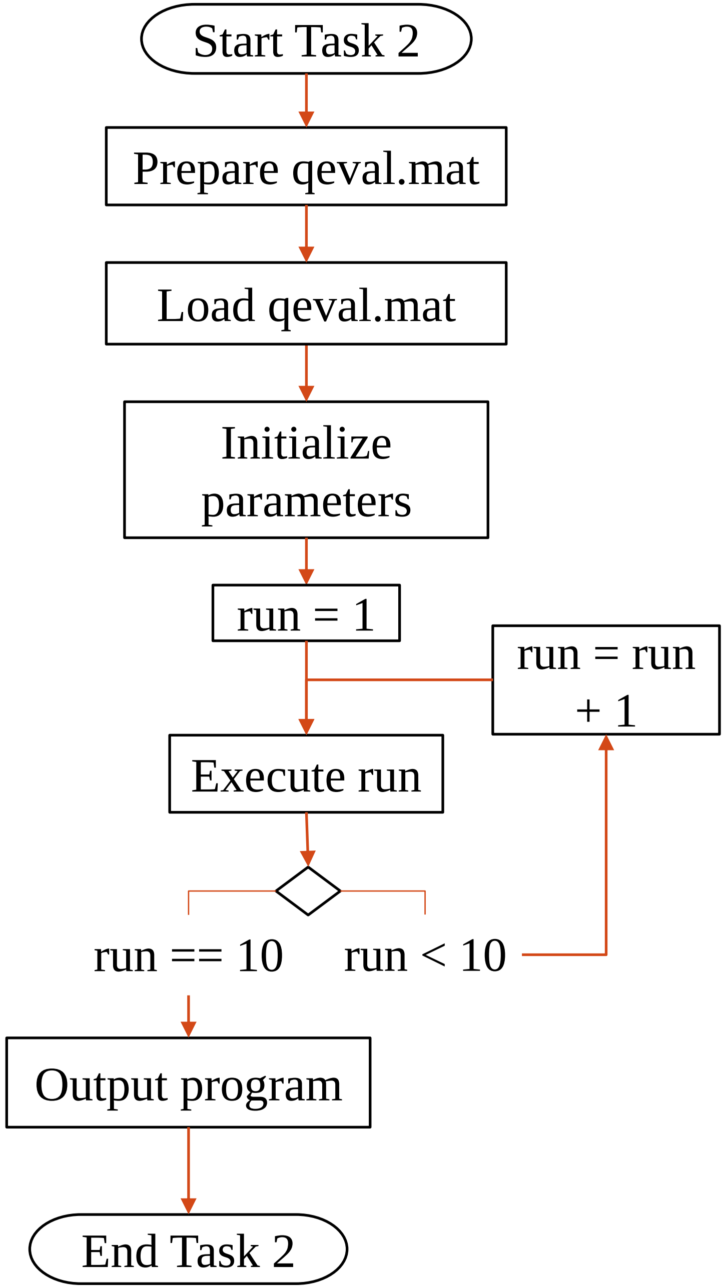
TASK II

WHAT TO DO?

*****Need to deal with unknown rewards**

1. Implement Q-learning algorithm in MATLAB.
2. Decide on your discount factor γ and exploration probability ϵ_k.
3. Complete report.

<u>Note: You can test its execution on your own by making up a qeval.mat file containing dummy sample values.</u>



ASSESSMENT

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 of the report**. This includes good report style, clarity, and conciseness
3. **Performance of your M-file program** for Task 2 in finding an optimal policy based on the reward function specified in file qeval.mat

SUBMISSION

WHAT TO SUBMIT?

A report (in a **PDF** file with **name** the report as: StudentNumber_RL.pdf) describing the implementation and the results. It must contain a cover page showing:

- Student's name
- Student number
- Student's email address
- Name of the module
- Project title

And the M-file programs.

ZIP A0000000X

Non-password protected!

RL_task1.m — **Task 1**

RL_main.m — **Task 2**

PDF A0000000X_RL .pdf — **Report**

APPENDIX - MATLAB COURSE

MATRIX	
Create 2 x 3 matrix	[1 2 3; 4 5 6]
Create 4 x 3 matrix of zeros	zeros(4, 3)
Find number of rows and columns of matrix A	size(A)
Get element at 1st row and 1st column of matrix A	A(1, 1)

REWARD	
Load ‘task1.mat’	load task1.mat
Create ‘qeval.mat’	save(‘qeval.mat’, ‘qevalreward’)

DATA	
Find the time taken of a block of code	tic % block of code toc
Display string with variable	disp([‘This is ‘ variable ‘variable.’])

TRAJECTORY PLOT

Plot coordinate x and y with arrows	• plot(x, y, ’^’); % action 1 • plot(x, y, ’>’); % action 2 • plot(x, y, ’v’); %, action 3 • plot(x, y, ’<’); % action 4
Set axis min and max	axis([0 10 0 10])
Format title	title(['Execution of optimal policy with associated reward = ‘ total_reward])
Show grid	grid on
Start grid from top left corner	set(gca,'YDir','reverse')

https://www.mathworks.com/help/matlab/getting-started-with-matlab.html

APPENDIX - RECAP

REWARD

Total reward for a state transition is given by:

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

where,

- R_t determines present value of future rewards

Rewards received k steps in the future is discounted by factor γ^{k-1}.

Small γ → Focus more on intermediate rewards for next the few steps.

Large γ → Take into account future rewards more strongly.

Q FUNCTION

'Worth' of actions at different states is given by:

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

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

Deterministic Transition

Expected return from taking action a at sate s at time step t by following action π

OPTIMAL POLICY

Optimal policy is the state transitions that maximize the Q-values

Slide 164-16

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

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

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 transitions.

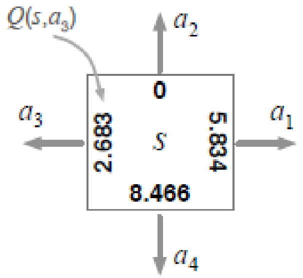
$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left(\underbrace{\overbrace{\boxed{r_{k+1}}}^{\text{Reward of action } a \text{ at state } s} + \gamma \max_{a'} Q_k\,(s_{k+1}, a')}_{\text{Estimate of } Q^*(s_k, a_k)} - Q_k(s_k, a_k) \right)$$

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

ϵ-GREEDY EXPLORATION

Initialize parameters

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

Select Action

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

Exploitation (with probability $1 - \epsilon_k$)

Exploration (with probability ϵ_k)

Apply Action

- Apply action a_k, receive reward r_{k+1}, then observe next state s_{k+1}

Update Q-value

- Update Q-function with:

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

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

THANKS FOR LISTENING