

Q-Learning in Grid World Environment

Team 09 - Homework 1

Shayan Meshkat Alsadat, Soham Karandikar, Aman Chandak, Abhinav Pillai

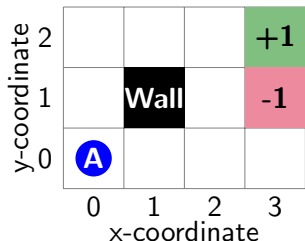
Reinforcement Learning in Robotics

October 4, 2025

Outline

- 1 Introduction and Environment Setup
- 2 Q-Learning Algorithm
- 3 Results and Analysis
- 4 Parameter Study
- 5 Conclusions
- 6 Implementation Details

Grid World Environment



Environment Specifications:

- **Grid Size:** 4×3
- **Starting Position for Agent:** $(0, 0)$
- **Goal:** $(3, 2)$ with reward $+1$
- **Penalty:** $(3, 1)$ with reward -1 or -100
- **Wall:** $(1, 1)$ - blocked
- **Step Cost:** -0.04 per move

Action Space:

- Up, Down, Left, Right
- Slip mechanics: 80% intended, 10% left, 10% right

Q-Learning Algorithm

Algorithm 1: Q-Learning Algorithm

```
1 Initialize  $Q(s, a) = 0 \forall s \in S, a \in A$ ,  $N$  steps;  
2 for each episode do  
3   Initialize state  $s = s_0 \in S$ ;  
4   while  $s \neq (3, 2) \vee s \neq (3, 1) \vee i \leq N$  do  
5      $\rho \leftarrow$  random number in  $[0, 1]$ ;  
6     if  $\rho < \epsilon$  then  
7       Choose random action  $a \in A$ ;  
8     else  
9        $a \leftarrow \arg \max_{a \in A} Q(s, a)$ ;  
10    end  
11     $s' \leftarrow$  take action  $a$ ,  $r \leftarrow$  receive reward;  
12    Update:  $Q(s, a) \leftarrow Q(s, a) + \alpha[r +$   
         $\gamma \max_{a'} Q(s', a') - Q(s, a)]$ ;  
13     $s \leftarrow s'$ ,  $i \leftarrow i + 1$ ;  
14  end  
15 end
```

Hyperparameters:

- Learning rate: $\alpha = 0.01$
- Discount factor: $\gamma = 0.9$
- Exploration rate: $\epsilon = 0.1$
- Max steps: 80
- Training episodes: 5000
- Test episodes: 1000

Q-Value Update Example - Step 1

Initial Q-Table (All Zeros)

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	0.000
(1,0)	0.000	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Parameters:

- $\alpha = 0.1$ (learning rate)
- $\gamma = 0.9$ (discount factor)
- $\epsilon = 0.1$ (exploration rate), Step cost:
 $r = -0.04$

Epsilon-Greedy Action Selection: Step 1: Generate random number

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.85$

Step 2: Action selection

if $\rho = 0.85 < \epsilon = 0.1$ then

| Choose random action $a \in A$

else

$a \leftarrow \arg \max_{a \in A} Q(s, a) =$
 $\arg \max_{a \in A} Q((0,0), a) = \arg \max_{a \in A} \{0, 0, 0, 0\}$

Since all Q-values are 0:

- All actions have equal Q-value
- Tie-breaking: choose randomly among best \rightarrow
Selected action: Right (random choice)

Q-Value Update Example - Step 1 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	0.000
(1,0)	0.000	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	0.000	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Update Formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

$$s = (0, 0), \quad a = \text{Right} \quad (1)$$

$$s' = (1, 0), \quad r = -0.04 \quad (2)$$

$$\max_{a'} Q(s', a') = 0 \quad (3)$$

$$Q((0, 0), \text{Right}) = 0 + 0.1[-0.04 + 0.9 \cdot 0 - 0] \quad (4)$$

$$= -0.004 \quad (5)$$

Current State: Agent moves to (1,0)

Q-Value Update Example - Step 2

Q-Table

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	0.000	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Current State: Agent at (1,0)

			+1
			-1
	A		

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.05$

Action selection:

if $\rho = 0.05 < \epsilon = 0.1$ **then**

 Choose random action $a \in A \rightarrow$ **Selected: Up**

else

$a \leftarrow \arg \max_{a \in A} Q((1, 0), a) =$
 $\arg \max_{a \in A} \{0, 0, 0, 0\}$

Agent randomly selects Up action

Q-Value Update Example - Step 2 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	0.000	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

$$s = (1, 0), \quad a = \text{Up} \quad (6)$$

$$s' = (1, 0), \quad r = -0.04 \quad (7)$$

$$\max_{a'} Q(s', a') = 0 \quad (8)$$

$$Q((1, 0), \text{Up}) = 0 + 0.1[-0.04 + 0.9 \cdot 0 - 0] \quad (9)$$

$$= -0.004 \quad (10)$$

Current State: Agent stays at (1,0)

Q-Value Update Example - Step 3

Q-Table

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Current State: Agent at (1,0)

			+1
			-1
	A		

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.08$

Action selection:

if $\rho = 0.08 < \epsilon = 0.1$ then

Choose random action $a \in A \rightarrow$ **Selected:**

Right

else

$a \leftarrow \arg \max_{a \in A} Q((1, 0), a) =$

$\arg \max_{a \in A} \{0, 0, 0, 0\}$

Agent randomly selects Right action

Q-Value Update Example - Step 3 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	0.000
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

$$s = (1, 0), \quad a = \text{Right} \quad (11)$$

$$s' = (2, 0), \quad r = -0.04 \quad (12)$$

$$\max_{a'} Q(s', a') = 0 \quad (13)$$

$$Q((1, 0), \text{Right}) = 0 + 0.1[-0.04 + 0.9 \cdot 0 - 0] \quad (14)$$

$$= -0.004 \quad (15)$$

Current State: Agent goes to (2,0)

Q-Value Update Example - Step 4

Q-Table

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Current State: Agent at (2,0)

			+1
	■		-1
		A	

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.5$

Action selection:

if $\rho = 0.5 < \epsilon = 0.1$ **then**

| Choose random action $a \in A \rightarrow$

else

| $a \leftarrow \arg \max_{a \in A} Q((2,0), a) =$
| $\arg \max_{a \in A} \{0, 0, 0, 0\}$

Tie break: agent randomly selects Down action

Q-Value Update Example - Step 4 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	0.000	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	-0.004	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

$$s = (2, 0), \quad a = \text{Down} \quad (16)$$

$$s' = (3, 0) \text{slip}, \quad r = -0.04 \quad (17)$$

$$\max_{a'} Q(s', a') = 0 \quad (18)$$

$$Q((2, 0), \text{Down}) = 0 + 0.1[-0.04 + 0.9 \cdot 0 - 0] \quad (19)$$

$$= -0.004 \quad (20)$$

Current State: Agent goes to (3, 0)

Q-Value Update Example - Step 5

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	-0.004	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Current State: Agent at (3,0)

			+1
	■		-1
			A

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.7$

Action selection:

if $\rho = 0.7 < \epsilon = 0.1$ **then**

| Choose random action $a \in A \rightarrow$

else

| $a \leftarrow \arg \max_{a \in A} Q((3,0), a) =$
| $\arg \max_{a \in A} \{0, 0, 0, 0\}$

Tie break: agent randomly selects Left action

Q-Value Update Example - Step 5 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	-0.004	0.000	0.000
(3,0)	0.000	0.000	0.000	0.000
...

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
...

$$s = (3, 0), \quad a = \text{Left} \quad (21)$$

$$s' = (2, 0), \quad r = -0.04 \quad (22)$$

$$\max_{a'} Q(s', a') = 0 \quad (23)$$

$$Q((3, 0), \text{Left}) = 0 + 0.1[-0.04 + 0.9 \cdot 0 - 0] \quad (24)$$

$$= -0.004 \quad (25)$$

Current State: Agent goes to (2, 0)

Q-Value Update Example - Step 6

Q-Table

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
...

Current State: Agent at (2,0)

			+1
	■		-1
		Ⓐ	

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.65$

Action selection:

if $\rho = 0.65 < \epsilon = 0.1$ **then**
| Choose random action $a \in A \rightarrow$

else

$a \leftarrow \arg \max_{a \in A} Q((2,0), a) =$
 $\arg \max_{a \in A} \{0, -0.004, 0, 0\}$

Tie break (between Up, Left, and Right): agent randomly selects Up action

Q-Value Update Example - Step 6 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	0.000	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
...

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
...

$$s = (2, 0), \quad a = \text{Up} \quad (26)$$

$$s' = (2, 1), \quad r = -0.04 \quad (27)$$

$$\max_{a'} Q(s', a') = 0 \quad (28)$$

$$Q((2, 0), \text{Up}) = 0 + 0.1[-0.04 + 0.9 \cdot 0 - 0] \quad (29)$$

$$= -0.004 \quad (30)$$


Current State: Agent goes to (2, 1)

Q-Value Update Example - Step 7

Q-Table

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
(2,1)	0.000	0.000	0.000	0.000
(2,2)	0.000	0.000	0.000	0.000
(3,1)	0.000	0.000	0.000	0.000
...

Current State: Agent at (2,1)

			+1
			-1

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.82$

Action selection:

if $\rho = 0.82 < \epsilon = 0.1$ **then**

 Choose random action $a \in A \rightarrow$

else

$a \leftarrow \arg \max_{a \in A} Q((2, 1), a) =$
 $\arg \max_{a \in A} \{0, 0, 0, 0\}$

Tie break: agent randomly selects Up action

Q-Value Update Example - Step 7 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
(2,1)	0.000	0.000	0.000	0.000
(2,2)	0.000	0.000	0.000	0.000
...

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
(2,1)	-0.004	0.000	0.000	0.000
(2,2)	0.000	0.000	0.000	0.000
...

$$s = (2, 1), \quad a = \text{Up} \quad (31)$$

$$s' = (2, 2), \quad r = -0.04 \quad (32)$$

$$\max_{a'} Q(s', a') = 0 \quad (33)$$

$$Q((2, 1), \text{Up}) = 0 + 0.1[-0.04 + 0.9 \cdot 0 - 0] = -0.004 \quad (34)$$

Current State: Agent goes to (2, 2)

Q-Value Update Example - Step 8

Q-Table

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	0.000
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
(2,1)	-0.004	0.000	0.000	0.000
(2,2)	0.000	0.000	0.000	0.000
...

Current State: Agent at (2,2)

		A	+1
			-1

$\rho \leftarrow$ uniform random number in $[0, 1] = 0.22$

Action selection:

if $\rho = 0.22 < \epsilon = 0.1$ **then**

 Choose random action $a \in A \rightarrow$

else

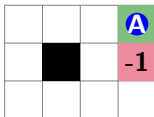
$a \leftarrow \arg \max_{a \in A} Q((2, 2), a) =$
 $\arg \max_{a \in A} \{0, 0, 0, 0\}$

Tie break: agent randomly selects Right action

Q-Value Update Example - Step 8 (continued)

Q-Table Before Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	0.000
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
(2,1)	-0.004	0.000	0.000	0.000
(2,2)	0.000	0.000	0.000	0.000
...



Current State: Agent goes to (3,2)

Q-Table After Update

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
(2,1)	-0.004	0.000	0.000	0.000
(2,2)	0.000	0.000	0.000	0.1
...

$$s = (2, 2), \quad a = \text{Right} \quad (35)$$

$$s' = (3, 2), \quad r = +1 \quad (36)$$

$$\max_{a'} Q(s', a') = 0 \quad (37)$$

$$Q((2, 2), \text{Right}) = 0 + 0.1[1 + 0.9 \cdot 0 - 0] = 0.1 \quad (38)$$

Q-Value Update Example - Final State

Final Q-Table

State	Up	Down	Left	Right
(0,0)	0.000	0.000	0.000	-0.004
(1,0)	-0.004	0.000	0.000	-0.004
(2,0)	-0.004	-0.004	0.000	0.000
(3,0)	0.000	0.000	-0.004	0.000
(2,1)	-0.004	0.000	0.000	0.000
(2,2)	0.000	0.000	0.000	0.1
(3,1)	0.000	0.000	0.000	0.000
(3,2)	0.000	0.000	0.000	0.000
...

Key Insights:

- Only visited state-action pairs get updated
- **Episode:** From start until agent reaches terminal state, e.g., reward +1 or -1.
- **Step:** Each action taken by the agent within an episode.
- Learning is incremental and experience-based

Training Structure: Runs, Episodes, and Steps

Algorithm 2: Q-Learning Training Structure

```
1 for run = 1 to num_runs do
2   Initialize  $Q(s, a) = 0 \forall s \in S, a \in A$ ;
3   for episode = 1 to max_episodes do
4     Initialize state  $s = s_{start}$ ;
5     step = 0;
6     while s not terminal AND step < max_steps do
7       Choose action  $a$  using  $\epsilon$ -greedy;
8       Execute action  $a$ , observe  $s'$  and  $r$ ;
9       Update:  $Q(s, a) \leftarrow$ 
           $Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ ;
10       $s \leftarrow s', \text{ step} \leftarrow \text{step} + 1$ ;
11    end
12    if episode % test_interval == 0 then
13      Evaluate current policy;
14    end
15  end
16 end
```

Hierarchical Structure:

- **Run (iteration):** Complete training process with independent initialization
- **Episode:** Single game from start to terminal state
- **Step:** Individual action taken within an episode

Our Configuration:

- **Runs:** 5 independent runs
- **Episodes:** 5000 per run
- **Steps:** Max 80 per episode
- **Testing:** Every 20 episodes

Agent Following Optimal Policy - Training

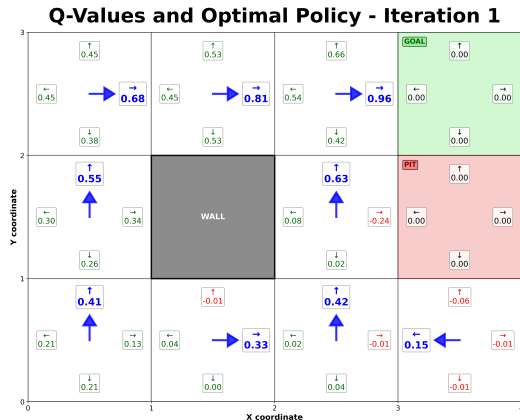


Figure: Optimal policy for the case with penalty of -1 .

Agent Following Optimal Policy - Training

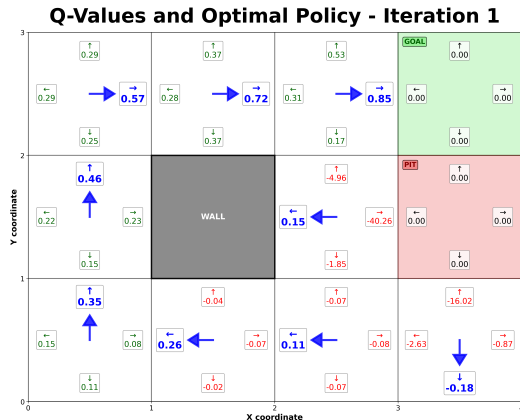


Figure: Optimal policy for the case with penalty of -200 .

Training Performance

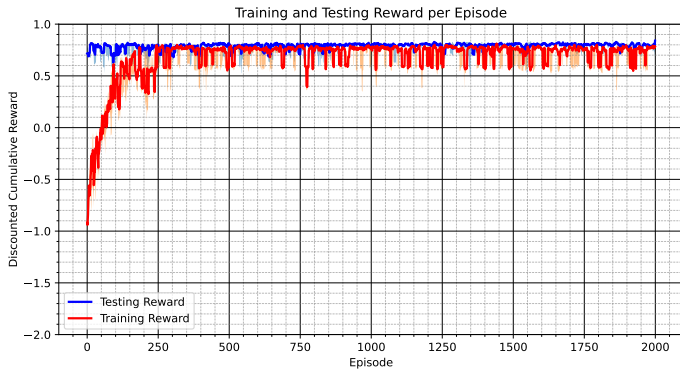


Figure: Training performance over 5000 episodes. Each curve represents the average reward over 5 independent runs. Shaded areas indicate percentile data. For the case with reward -1 .

Training Configuration:

- 5 independent runs per parameter setting (seed 0 to 4)
- 5000 training episodes per run
- Testing every 100 episodes
- Performance measured by average testing reward

Training Performance

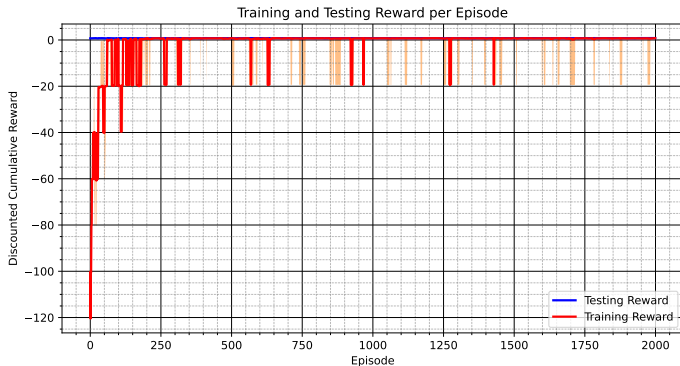


Figure: Training performance over 5000 episodes. Each curve represents the average reward over 5 independent runs. Shaded areas indicate percentile data. For the case with a reward of -200 .

Training Configuration:

- 5 independent runs per parameter setting (seed 0 to 4)
- 5000 training episodes per run
- Testing every 100 episodes
- Performance measured by average testing reward
- We obtain the same average reward of ≈ 0.84 , but with higher fluctuations due to high negative reward

Parameter Study Setup

Hyperparameter Ranges:

Learning Rate (α):

- 0.01
- 0.1
- 0.5
- 0.9

Discount Factor (γ):

- 0.1
- 0.3
- 0.9
- 0.95

Exploration Rate (ϵ):

- 0.01
- 0.1
- 0.3
- 0.5

Study Configuration:

- 5000 training episodes per run
- Testing every 100 episodes
- Performance measured by average testing reward

Parameter Study Results - Learning Rate

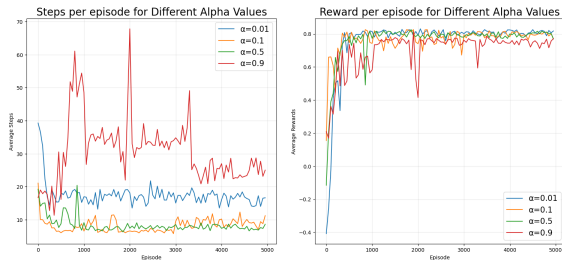


Figure: Training performance for different learning rates with a discount factor of 0.95, and exploration rates gradually decreased from 1. Each curve represents the average reward over 5 independent runs. The plot for steps (on the left) shows the number of steps until the termination condition for that episode is met.

How the learning rate affects learning?

- The learning rate directly affects how quickly the agent updates its knowledge: higher values allow rapid adjustment but can cause large, erratic fluctuations as seen in the plots, whereas lower values promote steady refinement and reliable but slower learning progress.

Parameter Study Results - Discount Factor

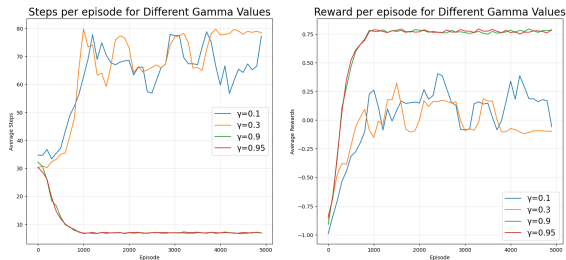


Figure: Training performance for different discount factors with a learning rate of 0.1, and exploration rates gradually decreased from 1. Each curve represents the average reward over 5 independent runs.

How the discount factor affects learning?

- The discount factor affects learning by determining how much the agent values future rewards. Higher values help the agent learn stable long-term strategies, whereas lower values make it focus on immediate rewards and may slow or limit learning, as shown.

Parameter Study Results - Exploration Rate

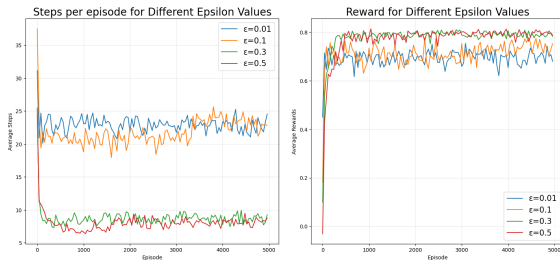


Figure: Training performance for different exploration rates with a learning rate of 0.1 and a discount factor of 0.95. Each curve represents the average reward over 5 independent runs.

How the exploration rate affects learning?

- The exploration rate affects learning by balancing exploration and exploitation; as shown in the plots, lower epsilon values lead to faster convergence and higher rewards, while higher values increase exploration but result in slower learning and reduced performance.

Convergence Study Setup

Hyperparameters:

Learning Rate (α):

- 0.1

Discount Factor (γ):

- 0.95

Exploration Rate (ϵ):

- 0.5

Convergence criteria: Q value

- Average absolute Q value change dropping below 0.001 over a sustained episode window for 5 independent runs is considered as Q value convergence.

Policy

- Average number of policy changes dropping below 0.1 over a sustained episode window for 5 independent runs is considered a policy convergence.

Study Configuration:

- 5000 training episodes per run
- Testing every 100 episodes
- Performance measured by average testing reward

Convergence Study Results

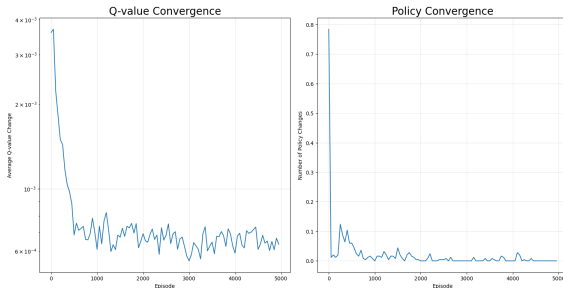


Figure: Average Q-value and policy changes per episode during training, with results averaged over 5 independent runs.

Does the Q-value converge first or the policy converge first?

- The policy converges first (near episode 500) because it stabilizes once the best action ranking for each state is identified, while Q-values keep making small updates even after the policy has stopped changing and converges near episode 700.

Comparison between the two reward cases

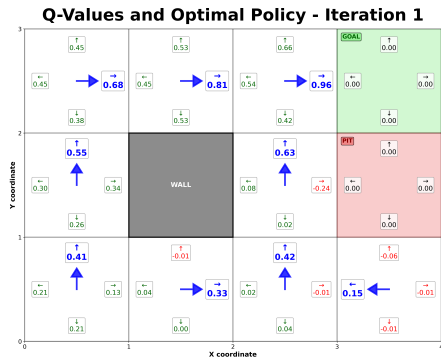


Figure: Optimal policy for the case with penalty of -1 .

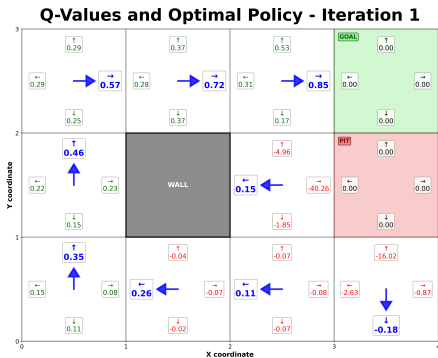


Figure: Optimal policy for the case with penalty of -200 .

Comparison between the two reward cases

How is the solution affected when the penalty is changed?

- In the low-penalty case (1), the agent often chooses direct paths toward the goal and does not strongly avoid state (2, 1) which is the state between the wall and the penalty.
- The low-penalty case optimal policy allows for riskier moves close to negative reward state, as the penalty does not outweigh gains from a shorter trajectory.
- With a high penalty (200), the policy aggressively avoids state (2, 1) west of the penalty and prefers safer alternatives, even if they require extra steps, reflecting strong aversion to large negative outcomes.
- In the high penalty case, the policy for state (2, 1) chooses actions that hit the wall rather than moving up or down, demonstrating an active strategy to avoid the risk of accidentally entering the high-penalty cell due to stochastic transitions.
- Comparing both cases shows that increasing the penalty shifts the agent's behavior from accepting small risks for a faster path to prioritizing safety and risk avoidance, which aligns with sensible decision-making in environments where negative outcomes are severe.

Key Findings

Algorithm Performance:

- Q-learning successfully learns an optimal policy for grid world navigation
- Converges to expected optimal path: avoid penalty, reach goal efficiently
- Handles slip mechanics effectively through exploration and learning

Parameter Insights:

- **Learning Rate:** Moderate values (0.1) provide best balance
- **Discount Factor:** High values (0.9) necessary for long-term planning
- **Exploration:** Balanced exploration (0.1) crucial for optimal learning

Environment Characteristics:

- Large penalty (-200) creates strong avoidance behavior
- Slip mechanics create stochasticity, requiring robust policy learning
- There might be several optimal policies

Key Implementation Features:

- **Slip Mechanics:** 80% intended action, 10% slip left, 10% slip right
- **Forbidden Transitions:** Wall boundaries and blocked cells
- **Reward Structure:** Goal (+1), Penalty (-200), Step cost (-0.04)
- **Policy Visualization:** Real-time animation of agent movement
- **Statistical Analysis:** Multiple runs with different random seeds

Code Organization

Project Structure:

Core Files:

- RL_q_learning.py
- environment.py
- policy_animation.py
- opt_policy_vis.py

Output Files:

- Training/Testing animations (GIF format)
- Static policy visualizations (PNG format)
- Performance data (Parquet format)
- Parameter study results and plots

Analysis Files:

- param_study.py
- save_results.py

Code Available at [GitHub Repository](#)