Reinforcement Learning

Assignment – 2

Q-Learning for Tic-Tac-Toe with Optimal Policy Evaluation

Submitted by:
Deep Das, Himal Rana, Vikram Singh
Roll No: *U23AI052, U23AI053, U23AI034*

Course: Reinforcement Learning
Institution: *Sardar Vallabhbhai National Institute of Technology, Surat*
Date: January 26, 2026

1 Aim

To implement a Q-Learning agent for the Tic-Tac-Toe environment and evaluate its learning performance by comparing the agent's actions against a precomputed optimal policy under different exploration rates.

2 Objectives

- To design a Tic-Tac-Toe environment suitable for reinforcement learning.
- To implement a tabular Q-Learning agent.
- To compute the optimal action for each reachable state.
- To analyze the effect of ϵ-greedy exploration.
- To evaluate learning using average reward and optimal action percentage.

3 Theory

Q-Learning is an off-policy, model-free reinforcement learning algorithm that learns the optimal action-value function $Q^*(s, a)$ using the update rule:

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

The ϵ-greedy strategy is used to balance exploration and exploitation. With probability ϵ, the agent selects a random action; otherwise, it selects the action with the highest Q-value.

4 Environment Description

The Tic-Tac-Toe environment consists of a 3×3 board represented as a vector of length 9:

- Empty cell: 0
- Player X: 1
- Player O: -1

The environment supports:

- Valid action generation

- Winner detection
- Terminal state identification
- Reward assignment

5 Optimal Policy Computation

An optimal policy is precomputed by exhaustively exploring all reachable game states. For each state, the set of optimal actions is stored and later used to evaluate whether the agent's chosen action is optimal.

This enables objective measurement of learning quality beyond win/loss outcomes.

6 Q-Learning Agent

The agent maintains a Q-table using a Python dictionary with state-action pairs.

6.1 Hyperparameters

- Learning rate (α): 0.5
- Discount factor (γ): 1.0
- Exploration rates (ϵ): 0, 0.1, 0.01

6.2 Sample Code

```
class QLearningAgent:
    def __init__(self, epsilon=0.1, alpha=0.5, gamma=1.0):
        self.q_table = {}
        self.epsilon = epsilon
        self.alpha = alpha
        self.gamma = gamma
```

7 Experimental Setup

The agent is trained and evaluated under the following configurations:

- Time steps: 800 and 2000
- Runs: 2000 and 4000

- Exploration rates: $\epsilon = \{0, 0.1, 0.01\}$

Performance metrics recorded:

- Average reward per time step
- Percentage of optimal actions taken

8 Results

The learning curves show that agents with non-zero ϵ values achieve better exploration and higher optimal action percentages over time. Purely greedy agents ($\epsilon = 0$) often converge prematurely to suboptimal policies.

Figure 1: Average Reward vs Time Steps

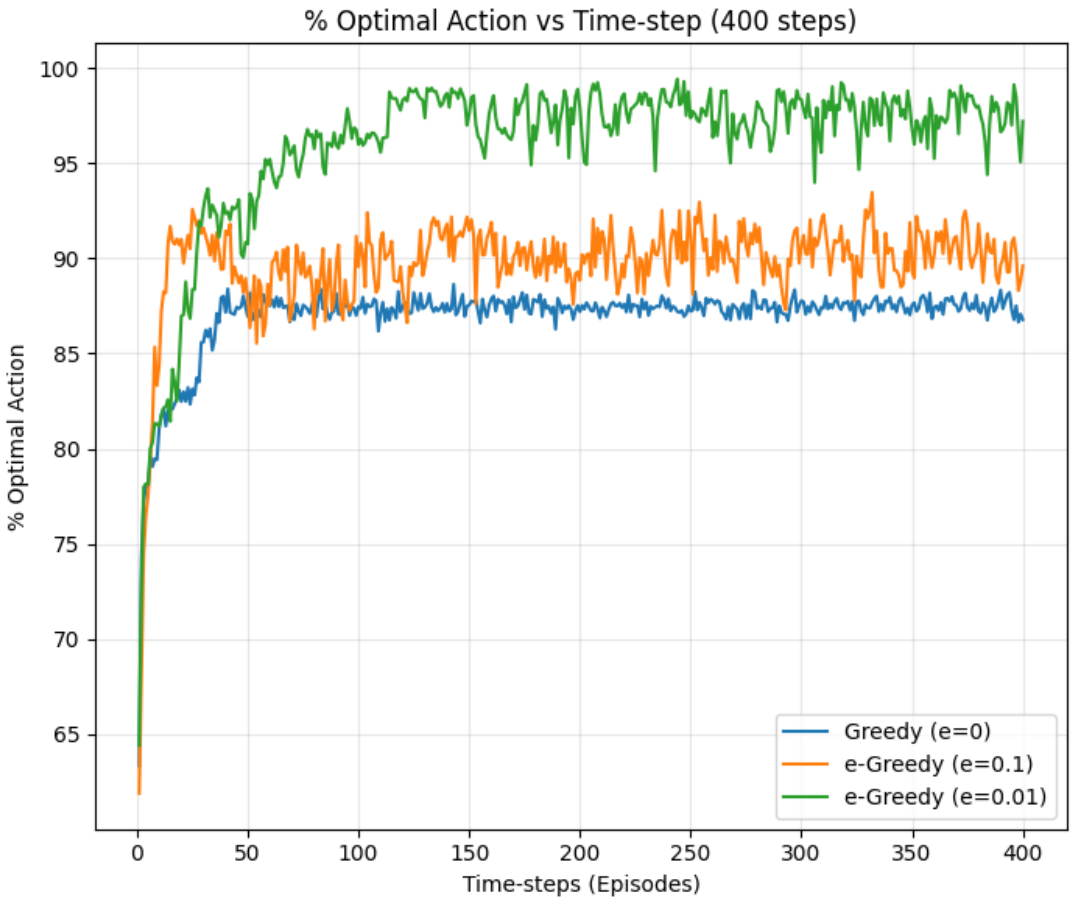

Figure 2: Percentage of Optimal Actions vs Time Steps

9 Analysis

Lower values of ϵ such as 0.01 balance exploration and exploitation effectively, resulting in smoother learning curves and higher optimal action rates compared to purely greedy behavior.

10 Conclusion

This experiment demonstrates that Q-Learning can successfully learn near-optimal Tic-Tac-Toe strategies. Evaluating agents using optimal action metrics provides deeper insight into learning quality than win rates alone.

11 References

1. Sutton, R. S., & Barto, A. G., *Reinforcement Learning: An Introduction.*