

# Q-Learning Agent Report

**Question 5 –** Write a Q-Learning agent in QLearningAgent.java by implementing the train() & extractPolicy() methods. Your agent should follow an  $\varepsilon$ -greedy policy during training (and only during training – during testing it should follow the extracted policy). Your agent will need to train for many episodes before the q-values converge. Although default values have been set/given in the code, you are strongly encouraged to play round with the hyperparameters of q-learning: the learning rate ( $\alpha$ ), number of episodes to train, as well as the epsilon in the  $\varepsilon$ -greedy policy followed during training. **(6 points)**

## Answer 5 –

### Summary for Q-Learning Agent Implementation

#### 1) Functions Modified:

- a) `train()`
- b) `extractPolicy()`
- c) Several helper methods for **strategic move selection**

#### 2) Detailed Implementation Analysis:

- a) **Training Method `train()`** - The training method implements a focused Q-learning algorithm with several features:

- (a) The agent uses an epsilon-greedy policy during training:

- 1. With probability epsilon, the agent explores (tries random moves)
    - 2. With probability 1 - epsilon, the agent exploits (chooses best-known move)
    - 3. Epsilon decays over time to reduce random exploration as learning progresses

- (b) **Episode-Based Learning**

- 1. Plays multiple episodes (games) to learn
    - 2. Resets environment at the start of each episode
    - 3. Continue until a terminal game state is reached

- (c) **Move Selection Strategy** - Two main selection approaches:

- 1. **Exploration:**

- a. Randomly selects moves while prioritizing defensive moves when exploring
    - b. Uses helper methods to find strategic moves

- 2. **Exploitation:**

- a. Selects moves with highest Q-values
    - b. Considers multiple strategic elements

- (d) **Reward Shaping**

- (i) Assigns different rewards based on game outcomes
    - (ii) Defensive moves: Small bonus (+5)

- (e) **Q-Value Update** Uses the Q-learning update rule:

$$Q(s,a) = (1 - \alpha) * Q(s,a) + \alpha * (\text{reward} + \gamma * \max(Q(s',a')))$$

This calculates expected future rewards

**b) Strategic Helper Methods** - The implementation includes several sophisticated helper methods:

(i) **findDefensiveMoves()** - Identifies moves that:

- a. Block opponent's potential wins
- b. Prevent fork opportunities
- c. Block two-in-a-row threats

(ii) **pickBestMove()** - Selects moves considering:

- a. Q-values
- b. Strategic board positions
- c. Immediate win detection
- d. Blocking critical moves

(iii) **wouldBlockTwoInARow() and wouldBlockForkOpportunity()**

- a. Advanced defensive strategies to prevent opponent's winning moves
- b. Analyzes board configurations to detect potential threats

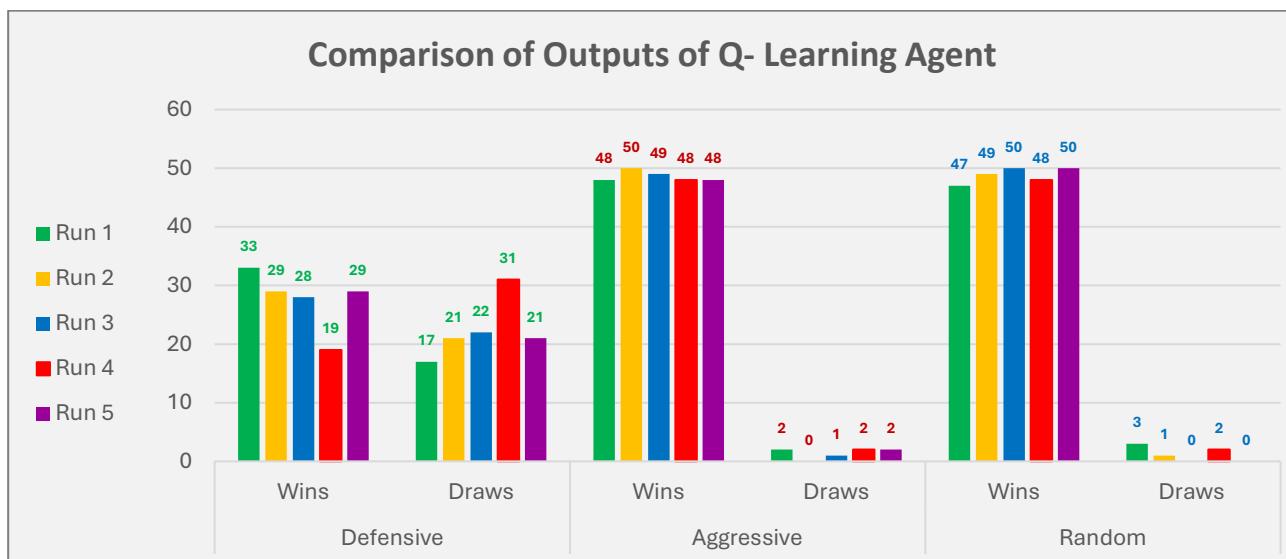
**c) Policy Extraction: extractPolicy()**

- i) Converts learned Q-values into a deterministic policy
- ii) For each game state, selects the move with the highest Q-value

**Question 6 –** Test your Value Iteration Agent against each of the provided agents 50 times and report on the results – how many games they won, lost & drew against each of the other rule-based agents. The rule-based agents are random, aggressive, defensive. (1 point)

## Answer 6 –

Iteration/ Agent	Defensive			Aggressive			Random		
	Wins	Losses	Draws	Wins	Losses	Draws	Wins	Losses	Draws
Run 1	33	0	17	48	0	2	47	0	3
Run 2	29	0	21	50	0	0	49	0	1
Run 3	28	0	22	49	0	1	50	0	0
Run 4	19	0	31	48	0	2	48	0	2
Run 5	29	0	21	48	0	2	50	0	0



The program was run for a total of 250 iterations split into 5 runs of 50 iterations each. As we can see from the tabular column, we obtain the best result during iteration 1 of the Value-Iteration Agent.

F2AI Final CW2 - C98L-cw2-factores-2024-2025/src/test/java/TestLearning.java - Eclipse IDE

File Edit View Insert Run Tools Window Help

Package Explorer JUnit Test Problems Javadoc Declaration Console

TestLearning.java

```
1 package org.junit.asserts;
2 import static org.junit.Assert.assertEquals;
3
4 import java.util.ArrayList;
5 import java.util.List;
6
7 import org.junit.Test;
8
9 public class TestLearning {
10     @Test
11     public void testAggressive() {
12         System.out.println("Against Aggressive Agent");
13         System.out.println("[" + LearningAgent.aggressive + "]");
14         System.out.println("[" + LearningAgent.random + "]");
15         System.out.println("[" + LearningAgent.defensive + "]");
16         System.out.println("[" + LearningAgent.passive + "]");
17         System.out.println("[" + LearningAgent.aggressive + "]");
18         System.out.println("[" + LearningAgent.random + "]");
19         System.out.println("[" + LearningAgent.defensive + "]");
20         System.out.println("[" + LearningAgent.passive + "]");
21         assertEquals(0, results[0]);
22     }
23
24     @Test
25     public void testRandom() {
26         System.out.println("Against Random Agent");
27         System.out.println("[" + LearningAgent.aggressive + "]");
28         System.out.println("[" + LearningAgent.random + "]");
29         System.out.println("[" + LearningAgent.defensive + "]");
30         System.out.println("[" + LearningAgent.passive + "]");
31         System.out.println("[" + LearningAgent.aggressive + "]");
32         System.out.println("[" + LearningAgent.random + "]");
33         System.out.println("[" + LearningAgent.defensive + "]");
34         System.out.println("[" + LearningAgent.passive + "]");
35         assertEquals(0, results[1]);
36     }
37
38     @Test
39     public void testDefensive() {
40         System.out.println("Against Defensive Agent");
41         System.out.println("[" + LearningAgent.aggressive + "]");
42         System.out.println("[" + LearningAgent.random + "]");
43         System.out.println("[" + LearningAgent.defensive + "]");
44         System.out.println("[" + LearningAgent.passive + "]");
45         System.out.println("[" + LearningAgent.aggressive + "]");
46         System.out.println("[" + LearningAgent.random + "]");
47         System.out.println("[" + LearningAgent.defensive + "]");
48         System.out.println("[" + LearningAgent.passive + "]");
49         assertEquals(0, results[2]);
50     }
51
52     @Test
53     public void testPassive() {
54         System.out.println("Against Passive Agent");
55         System.out.println("[" + LearningAgent.aggressive + "]");
56         System.out.println("[" + LearningAgent.random + "]");
57         System.out.println("[" + LearningAgent.defensive + "]");
58         System.out.println("[" + LearningAgent.passive + "]");
59         System.out.println("[" + LearningAgent.aggressive + "]");
60         System.out.println("[" + LearningAgent.random + "]");
61         System.out.println("[" + LearningAgent.defensive + "]");
62         System.out.println("[" + LearningAgent.passive + "]");
63         assertEquals(0, results[3]);
64     }
65 }
```

Problems Javadoc Declaration Console

terminated: TestLearning [Util] C:\Users\ARIF\Downloads\advice\justopenfile.hotspotge.ful

Running tests...

Playing move: P1Playing move: OCx2x1

[x][x][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing move: X(2,0)

[x][x][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing move: X(1,1)

[x][x][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing move: OCx2x1

[x][x][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing move: X(0,0)

[x][0][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing move: X(0,0)

[x][0][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing move: X(0,0)

[x][0][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing move: X(1,0)

[x][0][0]

[x][0][0]

[x][0][0]

[x][0][0]

Playing defensive move

Playing move: OCx2x2

[x][0][0]

[x][x][0]

[x][0][0]

[x][0][0]

Playing move: X(2,0)

[x][0][0]

[x][0][0]

[x][0][0]

[x][0][0]

X won

Score: 25 Wins: 25 Losses: 0 Drawn: 25