Implementing the Tic-Tac-Toe Game Using Reinforcement Learning

Tumpa Jalua

M.Tech,IIT Madras

September 5, 2023

Abstract

- We built an agent that learned to play the game of Tic-Tac-Toe.
- Three strategies are compared: self-play, the first player as a computer plays with a random player, and the first player as a random player plays with a computer player.
- Using the Q learning algorithm to train and test the three opponents.
- Calculate the winning probability of both players in these three strategies game.

Introduction

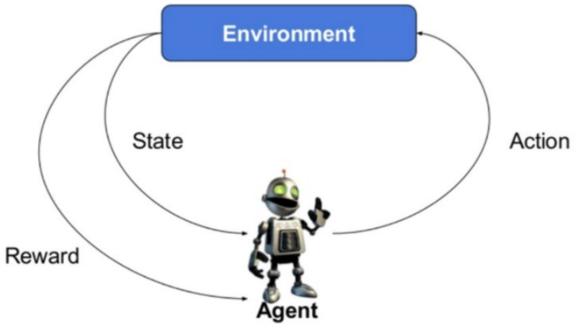
- Reinforcement learning is a step-by-step machine-learning process.
- After each step, the machine receives a reward that is reflected. Whether the step was good or bad in terms of achieving the target goal.
- By exploring its environment and exploiting the most rewarding steps, it learns to choose the best action at each stage.

Methodology

- In this game, we used a 3x3 board, comprising a total of 9 cells.
- One player plays Xs and the other plays Os.
- The board will be represented with symbols: 0 for available positions, 1 for Player 1's moves, and -1 for Player 2's moves.
- The outcome of each game, i.e., whether it results in a win, loss, or tie, and assign corresponding rewards of 1, -1, or 0, respectively

RL component

- State
- Environment
- Agent
- Action
- Reward



Q-Learning

- Exploration rate: The probability that our agent will explore the environment rather than exploit it.
- Exploitation rate: This involves the RL agent selecting actions that it believes will yield the highest immediate reward based on its current knowledge. Exploitation aims to maximize short-term gains.
- Epsilon greedy strategy: To balance exploration and exploitation by choosing between exploration and exploitation randomly.
- Q-learning algorithm: Q-learning is a reinforcement learning algorithm that learns the values of a function Q(s, a) to find an optimal policy. The values of the function Q(s, a) indicate how good it is to perform a certain action in a given state.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a' \in A} Q(s_0, a')\right) \tag{1}$$

Hyperparameter

- Learning rate: The agent updates its estimates incrementally as new experiences occur.
- Exponential decay epsilon: Reducing the learning rate over time.
- Linear decay epsilon: Gradually reduce the exploration rate over time

Action Selection

- Available position on the game board.
- Random Action Selection (Exploration): We generate a randomly generated number between 0 and 1. If this number is less than or equal to epsilon, then the agent will choose its next action via exploration,i.e. it will choose a random action from the available position.
- Highest Expected Reward (Exploitation): We generate a random number between 0 and 1. If this number is greater than epsilon, then the agent will choose its next action via exploitation, i.e. it will choose the action with the highest Q-value for its current state from the Q-table.

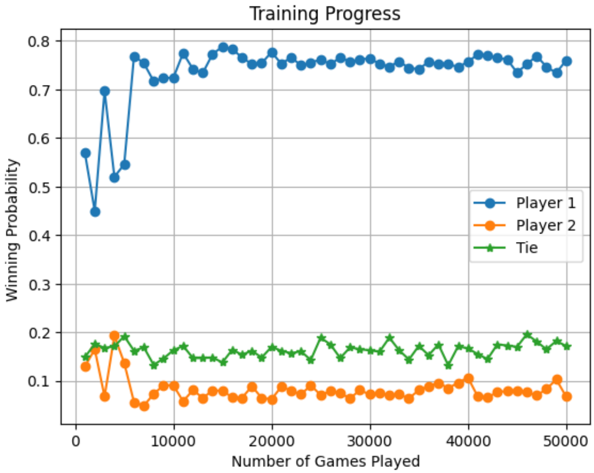
Result

Two random players play each other:

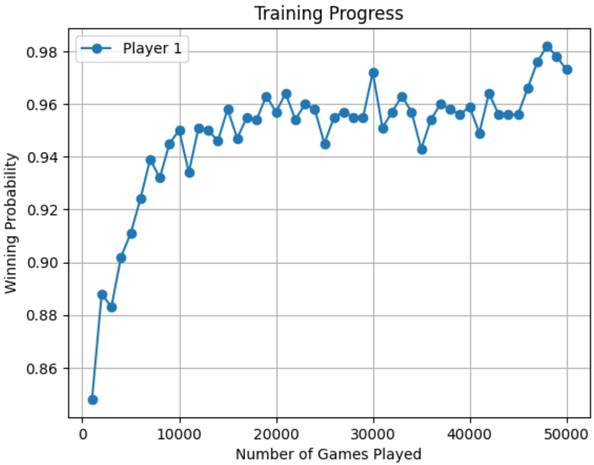
- Player 1 (RandomPlayerP1): Winning Probability: 57.40%
- Player 2 (RandomPlayerP2): Winning Probability: 43.60%

RL training

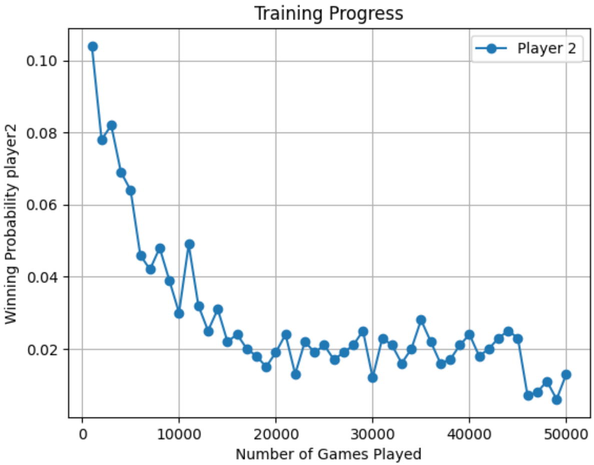
Constant exploration rate of both players



Exploration rate greater 1 of player1



Exploration rate greater 1 of player2



Exponential decay rate of three variants game

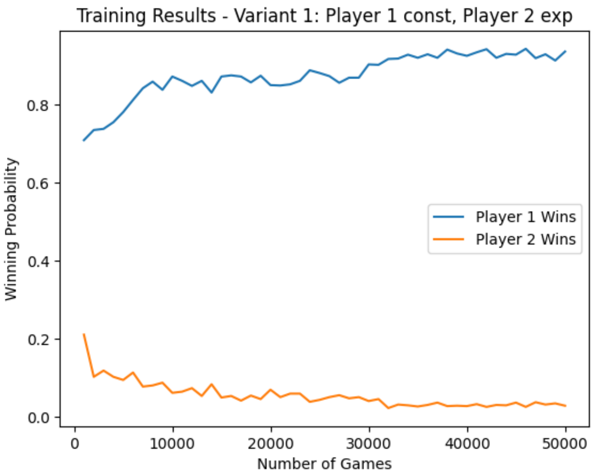


Figure: p1 constant and p2 decay

Continue

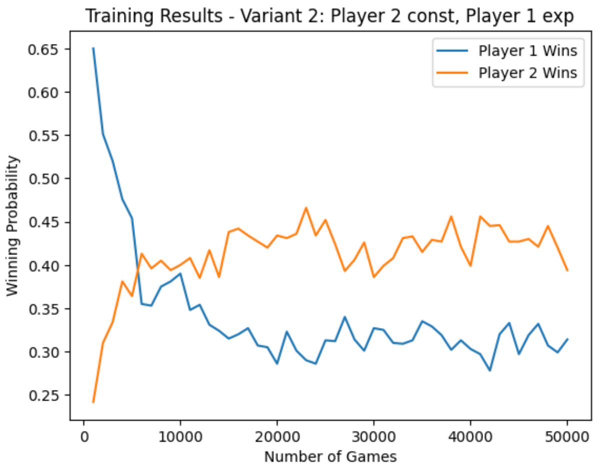


Figure: p2 decay and p1 constant

Continue

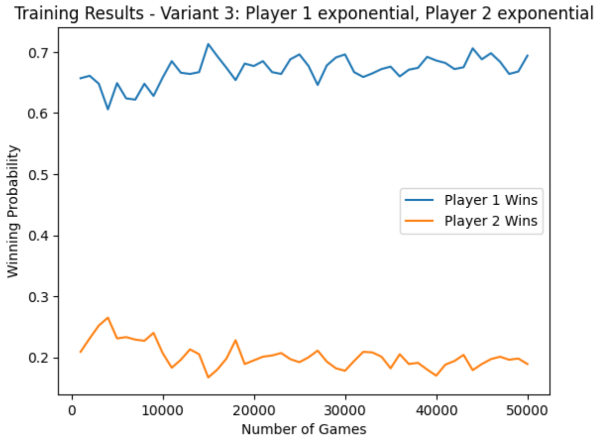


Figure: p1 and p2 decay

Linear decay rate of three variants game

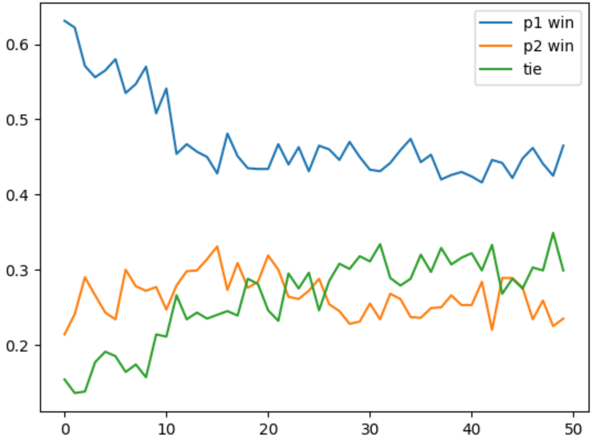


Figure: p1 constant and p2 decay

Continue

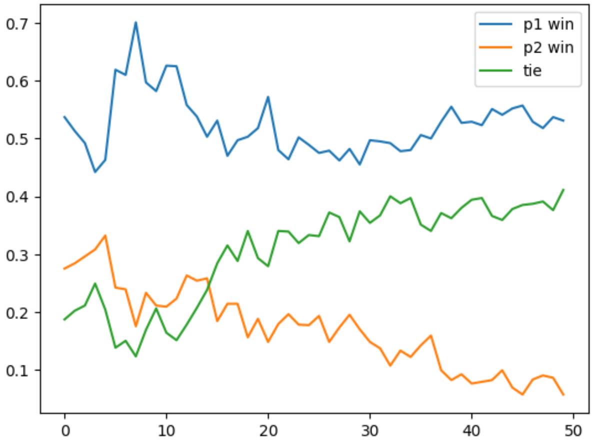


Figure: p2 decay and p1 constant

Continue

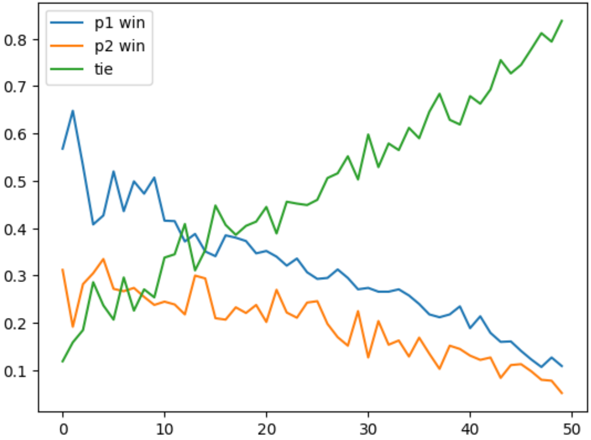


Figure: p1 and p2 decay

After training computer as p1 and Random player as p2

Test 1 - player trained with constant epsilon

- Player 1 (Computer): Winning Probability: 96.90%
- Player 2 (RandomPlayer): Winning Probability: 3.10%
- Tie Probability: 0.00%

Test 2 - one player trained with decaying epsilon

- Player 1 (Computer): Winning Probability: 98.40%
- Player 2 (RandomPlayer): Winning Probability: 0.00%
- Tie Probability: 1.60%

Test 3 - both players trained with decaying epsilon

- Player 1 (Computer): Winning Probability: 99.10%
- Player 2 (RandomPlayer): Winning Probability: 0.00%
- Tie Probability: 0.90%

After training human as p1 and computer as p2

Test 1 - player trained with constant epsilon

- Player 1 (RandomPlayer): Winning Probability: 49.20%
- Player 2 (Computer): Winning Probability: 46.00%
- Tie Probability: 4.80%

Test 2 - one player trained with decaying epsilon

- Player 1 (RandomPlayer): Winning Probability: 49.50%
- Player 2 (Computer): Winning Probability: 46.40%
- Tie Probability: 4.10%

Test 3 - both players trained with decaying epsilon

- Player 1 (RandomPlayer): Winning Probability: 52.30%
- Player 2 (Computer): Winning Probability: 44.10%
- Tie Probability: 3.60%

Conclusion

- RL often involves a lot of trial and error. The agent tries different actions, observes their outcomes, and adjusts its policy accordingly. Through this iterative process, it learns to make better decisions over time.
- We conclude that in a tic-tac-toe game, player 1 has an advantage when he plays first, and the winning probability is high with respect to player 2.
- When the first player is a computer and the second player is a random player, we see that the winning probability is averaged at 98
- Also, when the first player is a human and the second player is a computer player, the winning probability is on average 52

References

1 R. Sutton and A. Barto. Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA:pp. 10-15, 156. 1998.

2 C. Watkins and P. Dayan. Q-learning. Machine Learning, 8(3): 279-292, 1992.

3 https://towardsdatascience.com/reinforcement-learning-implement-tictactoe-189582bea542