TicTacToe Game Implementation Using Reinforcement Learning

Tumpa Jalua

IIT Madras

August 23, 2023

Introduction

- The project aims to develop a fully functional Tic-Tac-Toe game on a 3x3 board, comprising a total of 9 cells.
- One player plays Xs and the other 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 main goal of the project is to determine the outcome of each game, i.e., whether it results in a win, loss, or a tie, and assign corresponding rewards of 1, -1, or 0, respectively.
- First, we train two agent play each other and save their policy.
- Second, upload their policy in addition to make the agent to play with human.
- Third,the human to play with computer.

Keywards

The main keywards of these TicTactoe game are

- **Agent**:An agent is the learner or decision-maker within the reinforcement learning framework.
- **Environment**:Agent observe environment and decide to take action which changes environment(it may also change on its own).
- **State**:The state represents the current configuration of the game board, including the positions of the player moves (symbols). It serves as the input to the decision-making process, helping the players determine their next moves based on the current game situation.
- **Action**:Setup the actions left,right,up and down.
- **Reward**:The reward is a feedback given to the player after completing the game .For winning the game reward is 1 ,loss the game reward is -1 and tie the game reward is 0.
- Policy:It is agent behavior function or simply agent's behavior. The policy is a mapping from perceived states of the environment to actions to be taken when in those states or simply it maps the action to state.

Setup State

- **Initialization:** In this game initialized vacant board with two players p1 and p2.Each player has play symbol when one player play action its player symbol filled the board in addition to board state will exists updated.
- **Board State:** The board state is defined as the current configuration of the game board. It represents the positions of the player moves (symbols) on the board.
- **Available position:** Available positions refer to the positions on the game board where a player can make a move. These positions are unoccupied and do not have any player symbols (neither X nor O) on them. The availability of positions changes after each player move, as the board state is updated with the chosen action.
- **Check Winner:** After each action taken by a player, it is essential to check whether the game has reached a terminal state, i.e., if a player has won, lost, or if the game has resulted in a tie. This check is performed to determine the winner of the game and assign appropriate rewards to the players.

Setup Player

- Choose Action based on Current Estimation: During gameplay, each player (agent) makes decisions on which action (move) to take. These decisions are based on the current estimations of the expected rewards for different actions. To strike a balance between exploring new moves and exploiting known strategies, an epsilon-greedy policy is used. The policy helps the players decide whether to choose a random action (exploration) or select the action with the highest expected reward (exploitation) based on prior knowledge.
- Record the State of the Game:Throughout the game, the players record the states they encounter. These states represent the board configurations at different points in the game.
- Update states-value estimation after each game based on the rewards received.
- Save and Load Policies:o facilitate long-term learning and retain the knowledge gained from previous games, the players can save and load their policies.

Importance Point

- **Exploration:** Agents work on gathering more information to make the best decision.
- **Exploitation:** Agents make the best decision based on current information.
- **Epsilon greedy policy:**The epsilon greedy policy chooses between exploration and exploitation by estimating the highest rewards. It determines the optimal action. It takes advantage of previous knowledge to choose exploitation, looks for new options, and select exploration.
- **Exploration Rate:**the exploration rate is a crucial component of the epsilon-greedy policy. A high exploration rate means that the agent is more likely to explore new moves and positions on the board, even if it already has some knowledge about optimal actions. As the exploration rate decreases, the agent becomes more inclined to exploit its current knowledge and choose actions that have resulted in higher rewards in the past.

Training model

During training the process for each player currently exists

- looking for the available positions.
- Choose action.
- update board state in addition to add the action to player states.
- Judge if reach at the end of the game in addition to gives reward accordingly.

Choose action

- **Player1:** player1 choose action in the following ways.
- First available position on the game board.
- **Random Action Selection (Exploration):** If a randomly generated number between 0 in addition to 1 currently exists less than or equal to player1 exploration rate,so player1 selects a random action from the available position.
- **Action Selection with Highest Expected Reward (Exploitation):**If the randomly generated number currently exists greater than the exploration rate player1 chooses action with the highest expected reward.

Continue choose action

- **Evaluating Expected Rewards:** For each available position, Player 1 creates a copy of the current game board and updates it with the chosen position. It then calculates the hash value of the resulting board state.
- Handling Missing Reward Value:If no value is found in the states-value estimations for the calculated board hash, Player 1 defaults to a reward value of 0.
- Maximizing Expected Reward: Finally, Player 1 selects the action that maximizes the expected reward among the available positions. This action is chosen as the move to be played on the game board.

Continue choose action

- **Player2:**Player 2's action selection follows a similar process to that of Player 1.
- Available positions.
- Player 2's action selection follows a similar process to that of Player 1.
- **Random Action Selection (Exploration):** If a randomly generated number between 0 and 1 is less than or equal to Player 2's exploration rate, it selects a random action from the available positions. This random exploration allows Player 2 to explore new moves and potential strategies.
- **Action Selection with Highest Expected Reward (Exploitation):**If the randomly generated number is greater than the exploration rate, Player 2 chooses an action with the highest expected reward based on its learned state-value estimations. This exploitation phase allows Player 2 to prioritize actions that have resulted in higher rewards in previous games.

Setup human vs Player

- In the human vs. player mode, actions will be chosen by the human player.
- Choose action: this action choose for humans
- Game states will be added.
- rewards will be backpropagated.
- The game will be reset for a new round.

Result

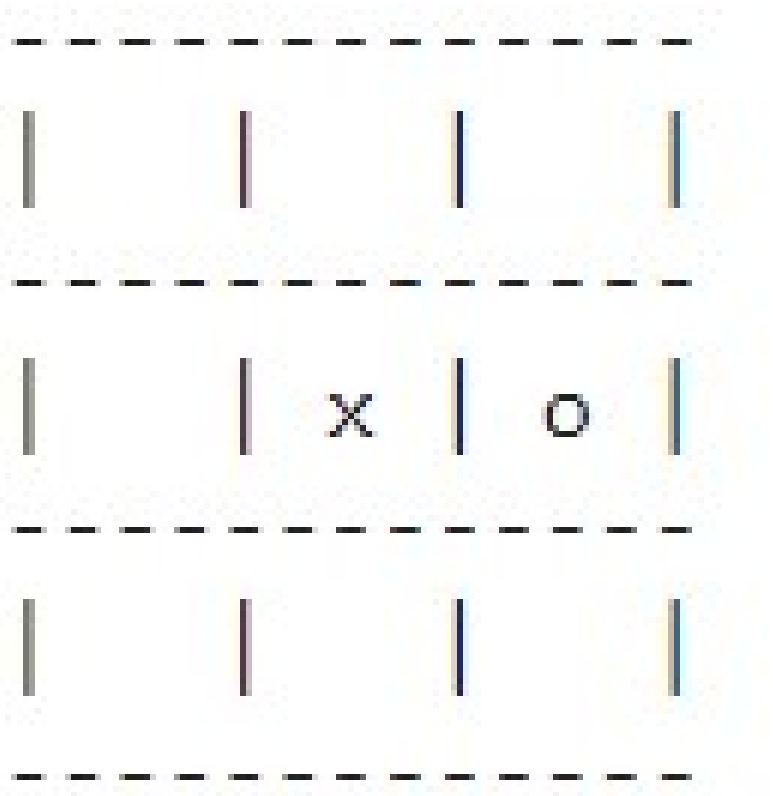


Figure: TicTacToe-step1

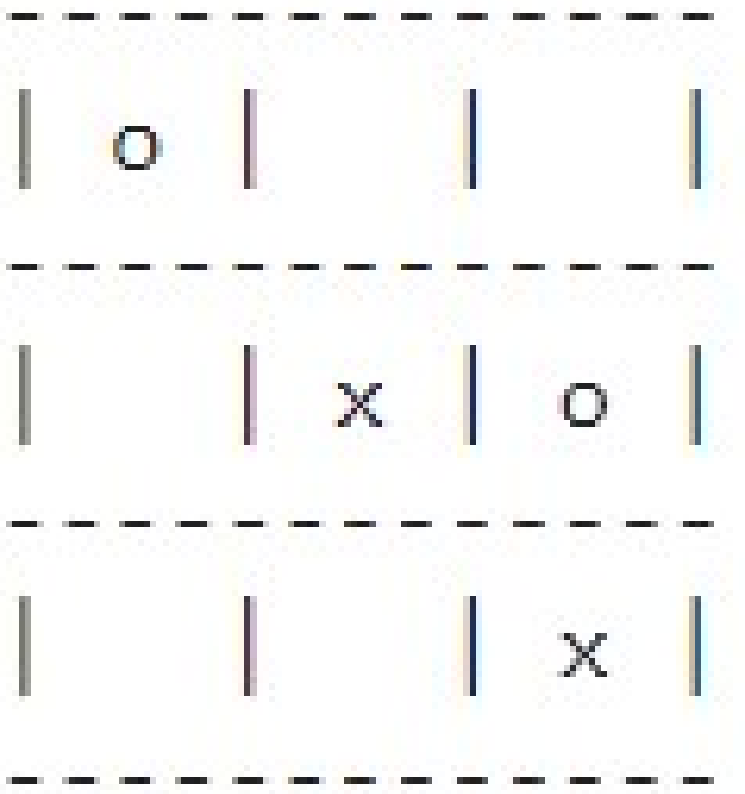


Figure: TicTacToe-step2

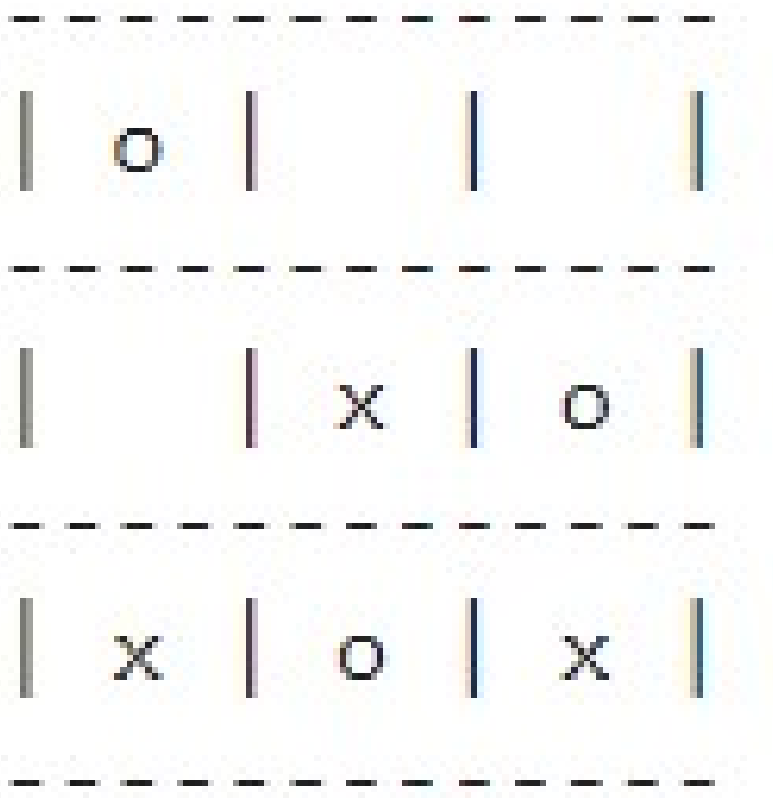


Figure: TicTacToe-step3

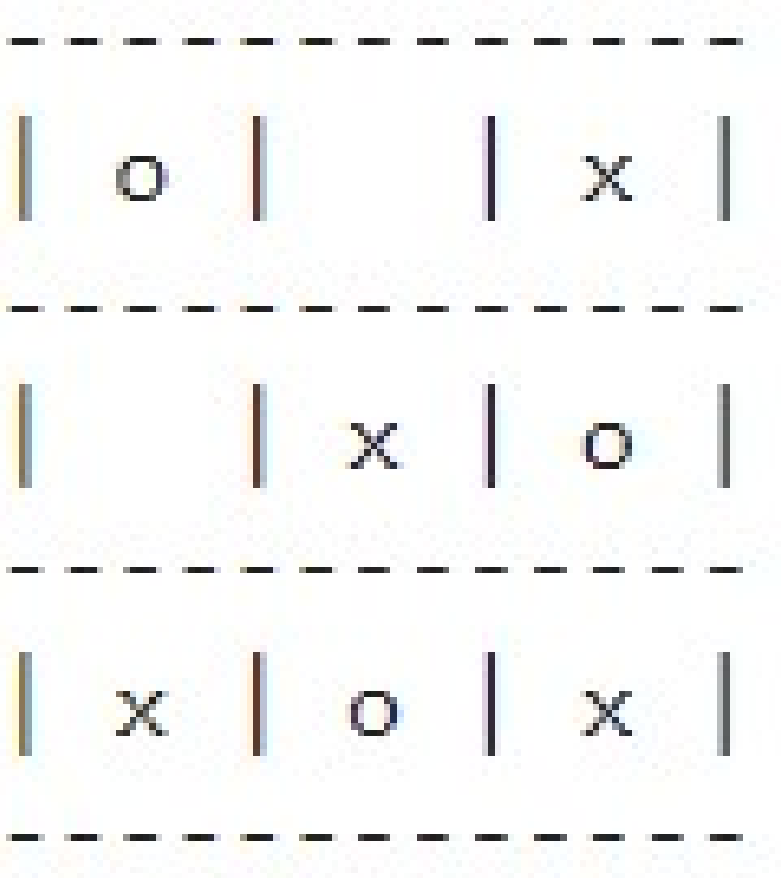


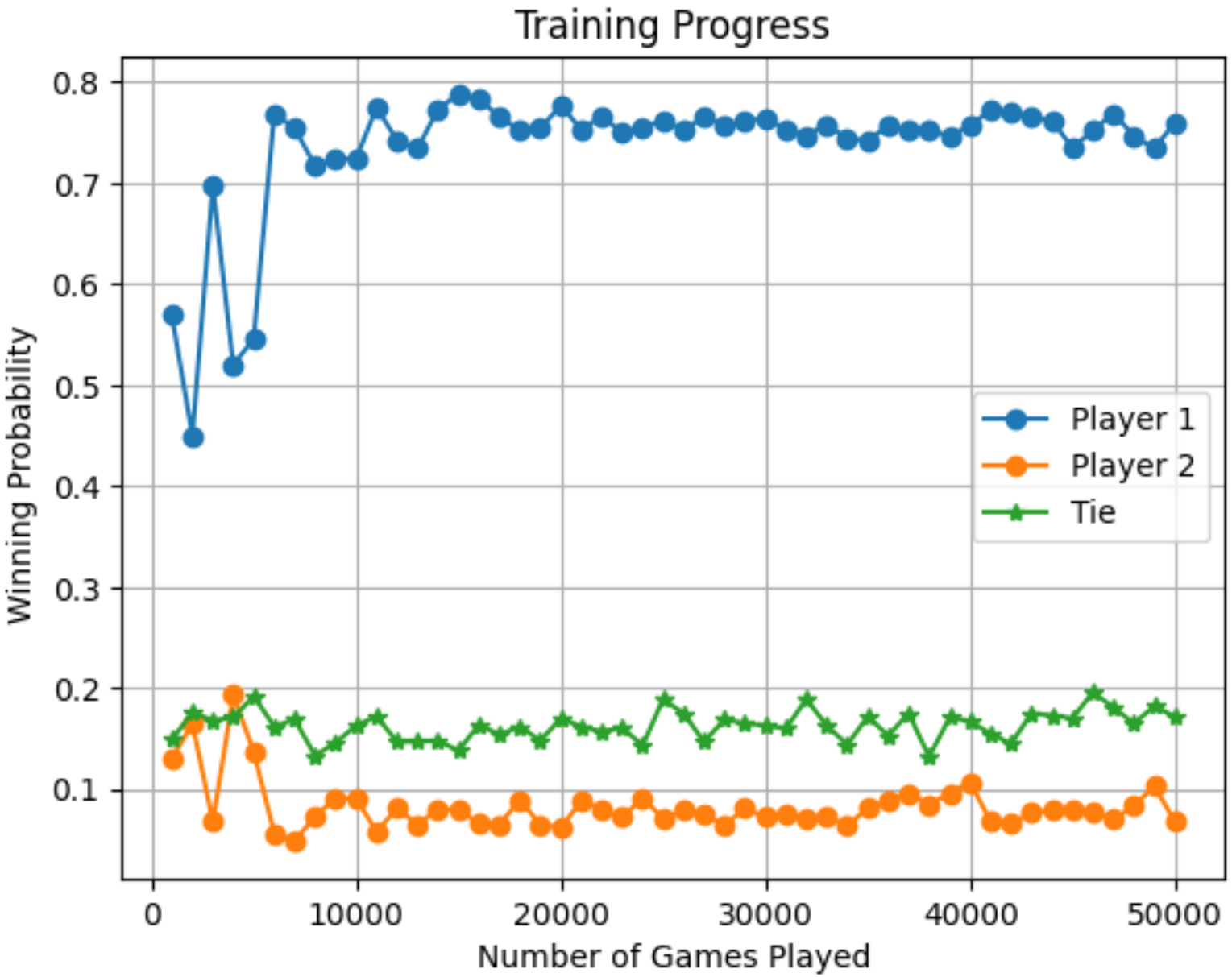
Figure: TicTacToe-step4

Figure: Player1 win

Output

No. of games	Player 1 win percentage.	Player2 win percentage	tie percentage
1 to 1000	78.30%	7.00%	14.70%
1001 to 2000	77.10%	7.30%	15.60%
2001 to 3000	75.50%	8.10%	16.40%
3001 to 4000	72.90%	7.10%	20.00%
4001 to 5000	74.10%	9.30%	17.40%
5001 to 6000	76.60%	7.90%	15.50%
45001 to 46000	74.70%	9.60%	15.70%
46001 to 47000	76.40%	9.00%	14.60%
47001 to 48000	76.10%	8.40%	15.50%
48001 to 49000	75.60%	10.80%	13.60%
49001 to 50000	77.00%	8.10%	14.90%

Player1 and player2 constant exploration rate



Output

No. of games	Player 1 Prob.	Player 2 Prob.	Game tie
1 to 1000	75.20%	6.40%	17.40%
1001 to 2000	76.40%	8.00%	15.60%
2001 to 3000	76.30%	7.30%	16.40%
3001 to 4000	77.50%	6.10%	16.40%
4001 to 5000	75.60%	6.90%	17.50%
5001 to 6000	73.70%	7.10%	19.20%
45001 to 46000	76.00%	6.90%	16.90%
46001 to 47000	78.40%	6.90%	14.70%
47001 to 48000	78.20%	7.80%	14.00%
48001 to 49000	74.70%	9.70%	15.60%
49001 to 50000	75.50%	6.90%	17.60%

Table: Training Results with exponential decay rate

exponential decay rate

Output when exploration rate greater than 1

No. of games	Player 1 Prob.	Player 2 Prob.	Game tie
1000	0.848	0.104	0.048
2000	0.888	0.078	0.034
3000	0.883	0.082	0.035
4000	0.902	0.069	0.029
5000	0.911	0.064	0.025
45000	0.956	0.023	0.021
46000	0.966	0.007	0.027
47000	0.976	0.008	0.016
48000	0.982	0.011	0.007
49000	0.978	0.006	0.016
50000	0.973	0.013	0.014

Table: Training Results

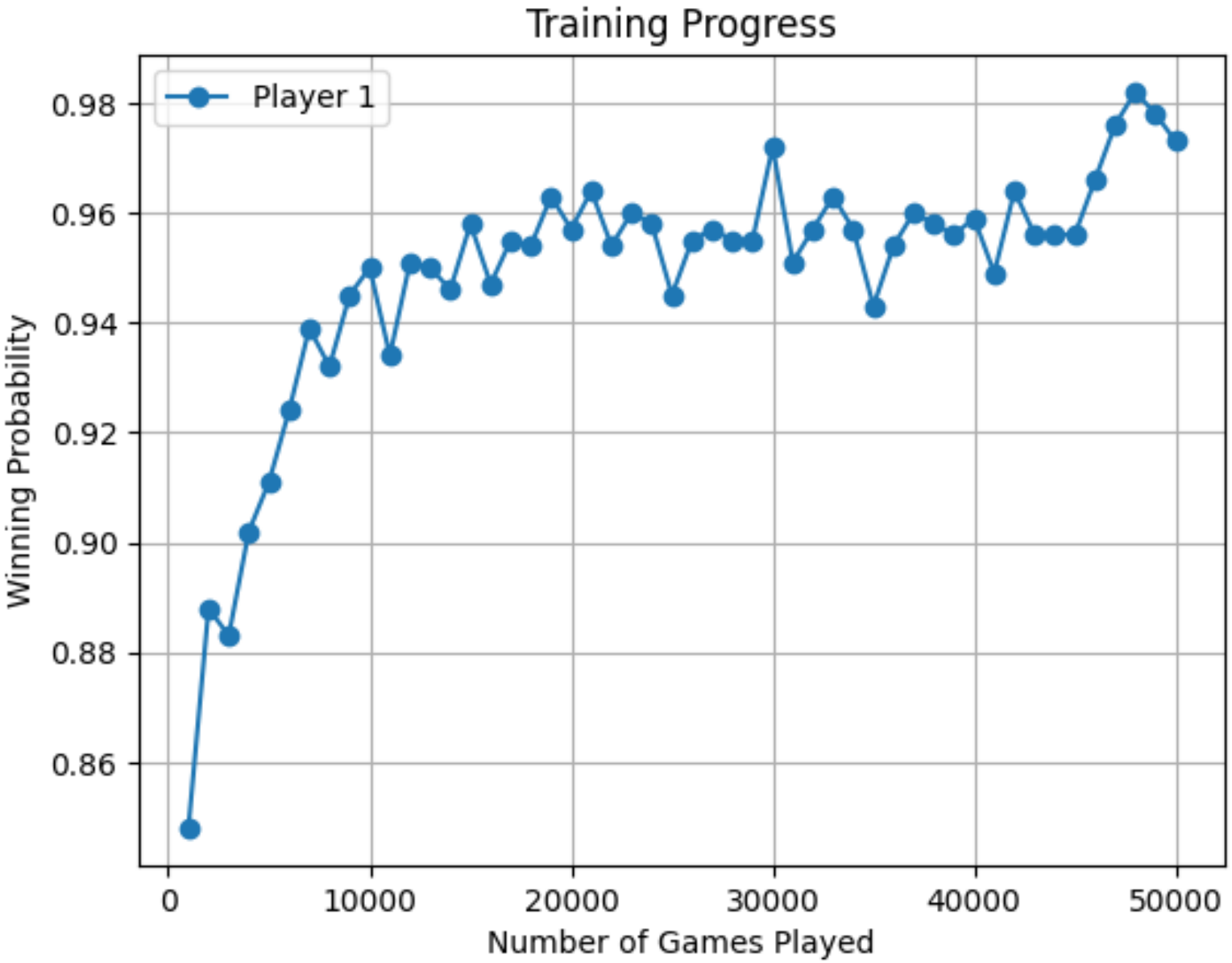
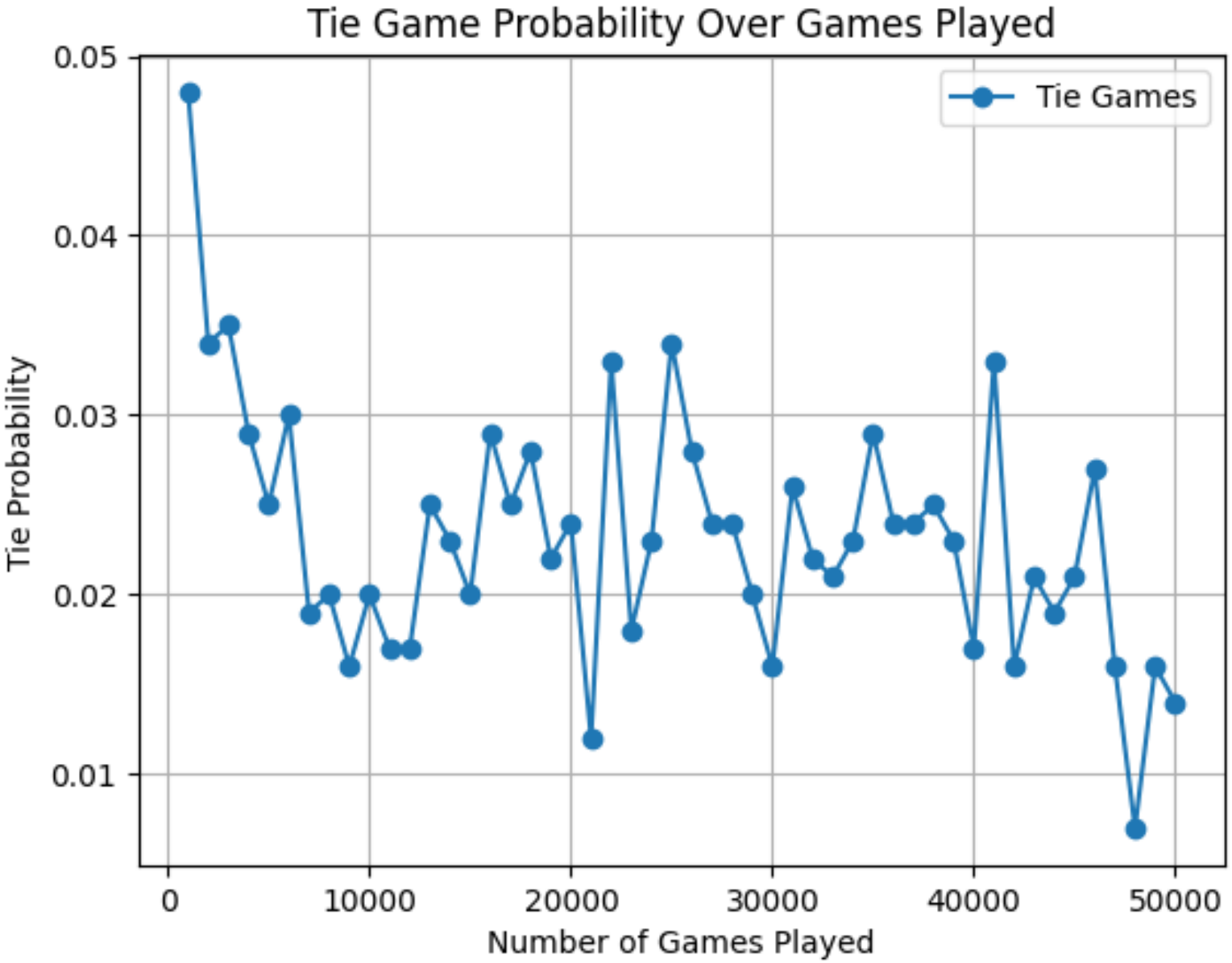
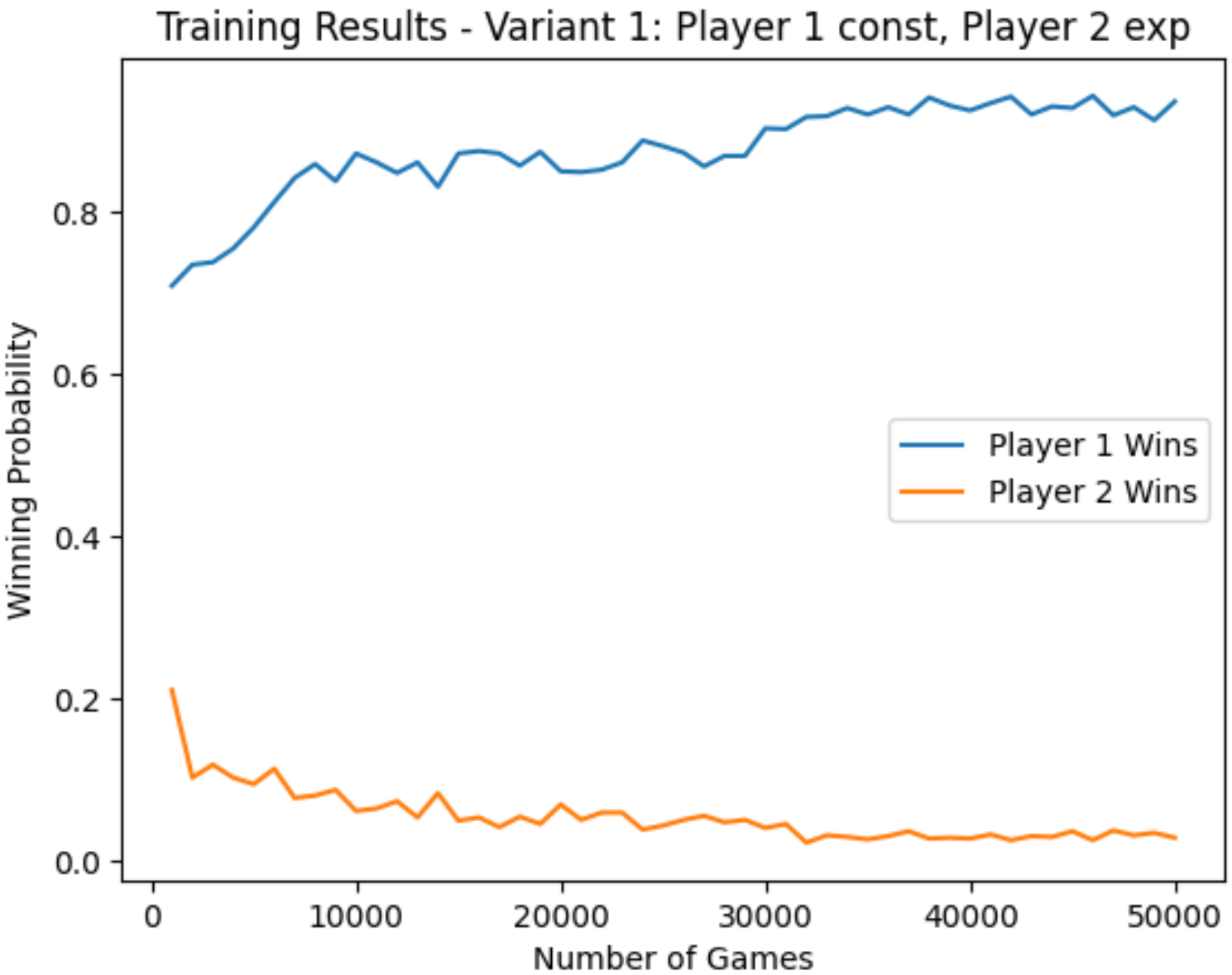


Figure: proportion of player2

Figure: probability of player2


Tie Game Probability Over Games Played
Tie Games
Tie Probability
0.05
0.04
0.03
0.02
0.01
0
10000
20000
30000
40000
50000
Number of Games Played

Three variant game



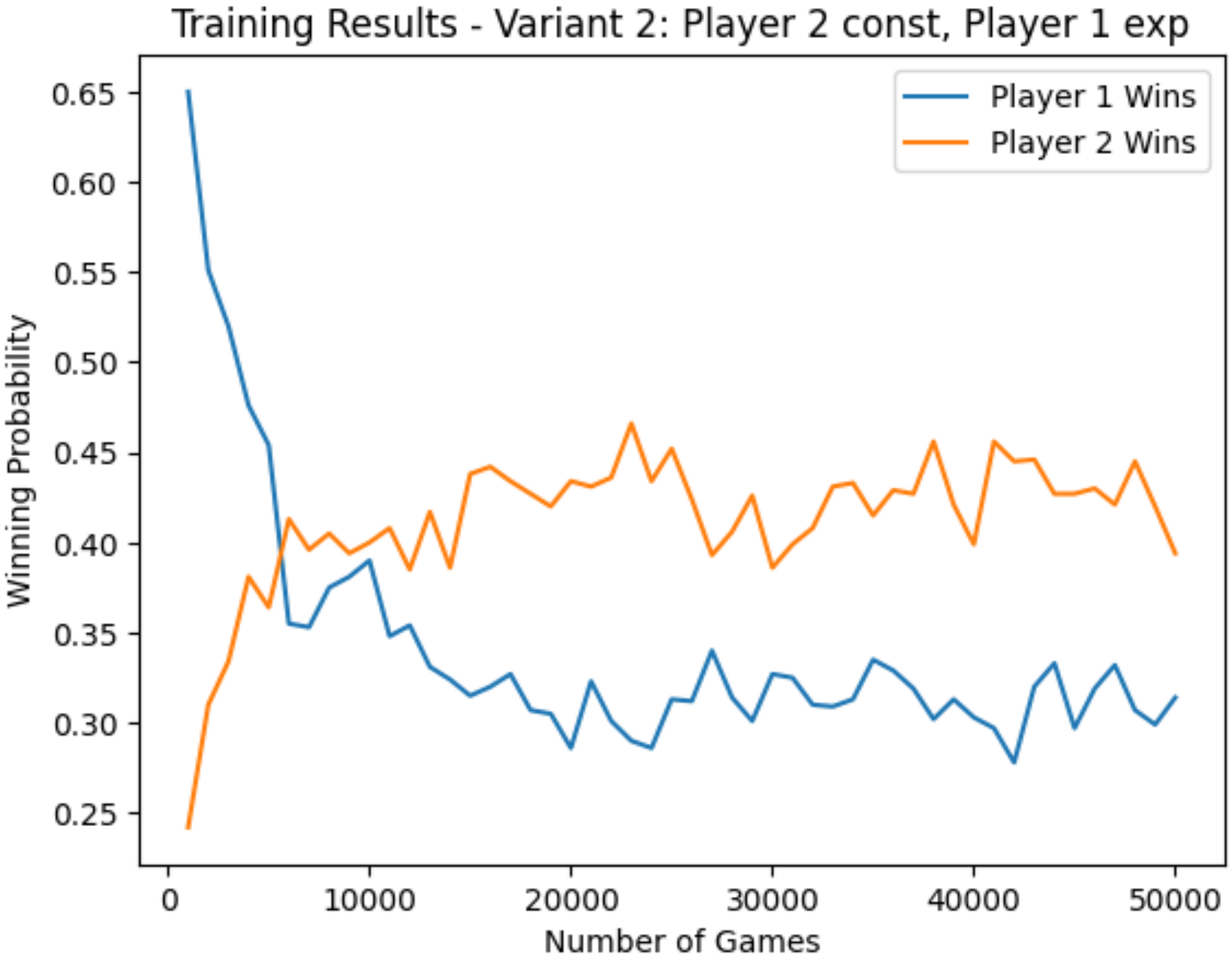


Figure: p2 decay and p1 constant

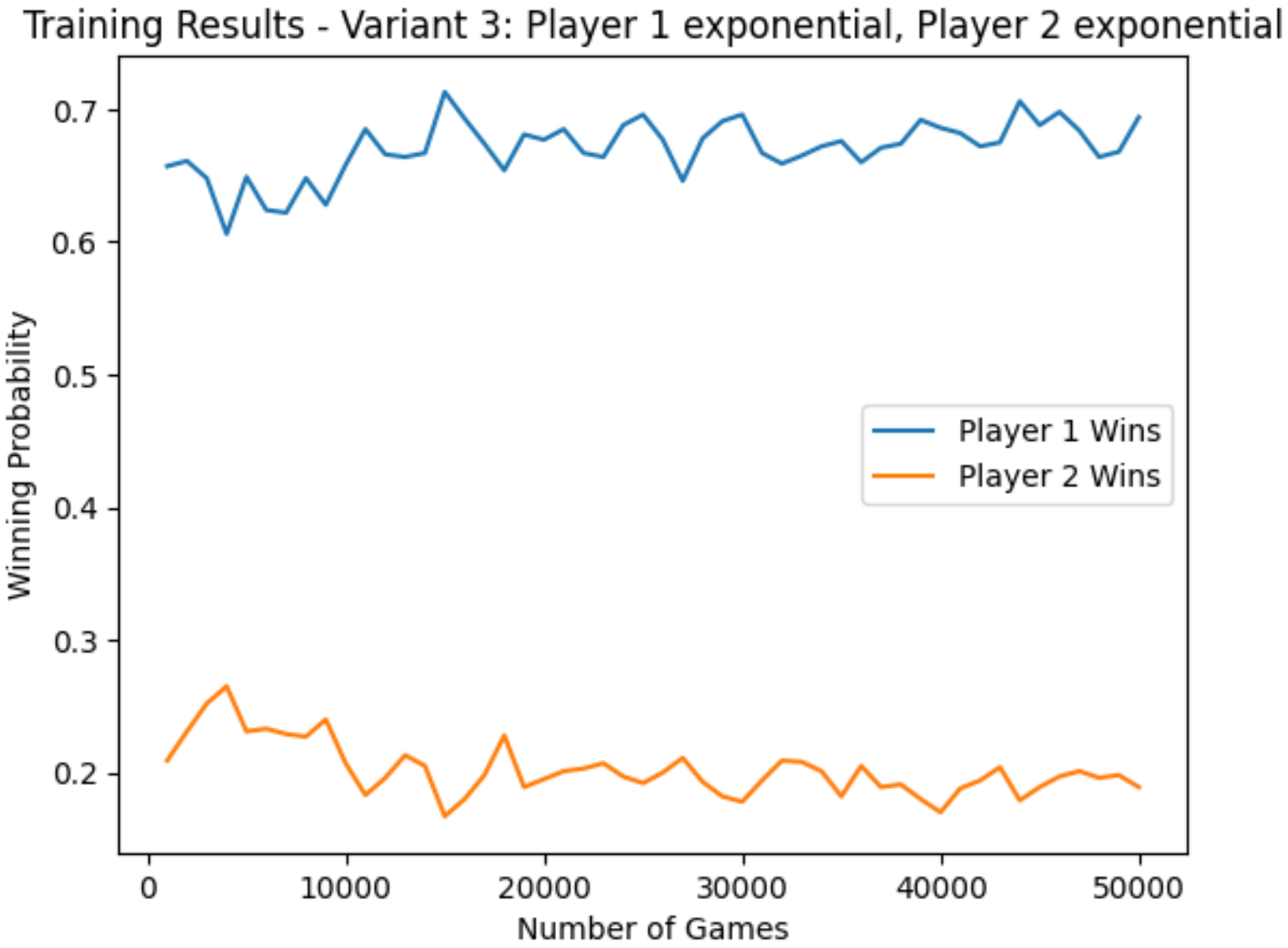


Figure: p1 and p2 decay

play2,human as p1 and selection random,p2 as computer

- 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%

play2,human as p2 and selection random,p1 as computer

- 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%

Conclusion

- Reinforcement learning(RL) is a amazingly powerful algorithm that employs a series of relatively simple steps to chained together to produce a form of artificial intelligence. These agents can be inherent qualities of any purpose-driven decision maker. The application and testing of the Q-learning algorithm for playing TicTacToe have been presented, showcasing its efficiency. As agents engage in an increasing number of games against opponents, their win rates increase while their loss rates decrease.Trained RL agent that Identify strategies for human to follow during the game.

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. G. Tesauro. Temporal difference learning and TD-Gammon. Communications of ACM, 38(3):pp. 58-68, 1995.[4] Figure source: Boulter.com/ttt/

- https://towardsdatascience.com/reinforcement-learning-implement-tictactoe-189582bea542
- https://deepanshut041.github.io/Reinforcement-Learning/notes/00$_I ntroduction_t o_r I$/

- https://saturncloud.io/blog/properly-set-up-exponential-decay-of-learning-rate-in-tensorflow/