IDENTIFICATION OF STRATEGIES IN MULTIPLAYER GAMES USING REINFORCEMENT LEARNING

Tumpa Jalua (MA22M024)

Under the Supervision of
Dr. Sivaram Ambikasaran
Dr. Sri Vallabha Deevi (Tiger Analytics)

Department of Mathematics
Indian Institute of Technology, Madras

May 15, 2024

Introduction

- In today's digital world, customer interactions with companies are primarily digital.
- Customer interactions with e-commerce companies can be modeled as two-player games.
- The customer is one player trying to find products/services at optimal prices and maximize discounts.
- The company is the other player trying to maximize revenue/profit from the customer, by influencing them to purchase.

Reinforcement Learning

- Reinforcement Learning (RL) [3] is a machine-learning model to train agents to play multi-step games.
- After a series of steps, the machine receives a reward that shows whether the series of steps were good or bad in terms of achieving the target goal.
- Reinforcement Learning is a suitable method to teach agents to play two-player games.
- By exploring its environment and exploiting the most rewarding steps, it learns to choose the best action at each stage.

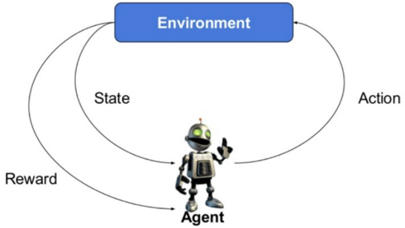


Figure 1: RL components [1].

Q-learning Algorithm

- The algorithm employs a Q-table of state-action pairs.
- The state space represents possible agent states, while the action space represents potential actions.
- The Q-table is a matrix storing the expected reward for each action in each state.

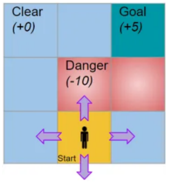


Figure 2: Example of Q-learning [2].

State	Actions			
	Action 1	Action 2	Action 3	Action 4
State 1	0.1	0.85	0.05	0.7
State 2	0.45	0.35	0.2	0.3
State 3	0.01	0.35	0.64	0.25

Table 1: State-Action Value.

State	Value
State i	0.15
State j	0.2
...	...

Table 2: State Value Function

Exploration-Exploitation and DQN

- **Exploration:** Agent chooses a random action to gather more information about the environment and update its understanding.
- **Exploitation:** Agents make the best decision based on current information.
- **Epsilon Greedy Strategy:** The epsilon-greedy strategy is a policy that handles the exploration/exploitation trade-off.
- **Deep Q-Network (DQN):** DQN refers to the neural network utilized within the Deep Q-learning framework to estimate the Q value or State value.
- During training, the DQN receives experience tuples containing the current state, action, reward, and next state.
- When the agent interacts with the environment, the generated tuples are stored in the replay buffer.
- This buffer is then used to train the DQN using an optimizer like Stochastic Gradient Descent or Adam optimizer.

Tic-Tac-Toe game using Q-learning

- We set up an empty board of size 3x3 comprising a total of 9 cells.
- Designated two players as Player 1 (P1) and Player 2 (P2). We assume P1 plays 'X' as the first player.
- The board is represented with symbols: '0' for available positions, '1' for Player 1's moves ('X'), and '-1' for Player 2's moves ('O').

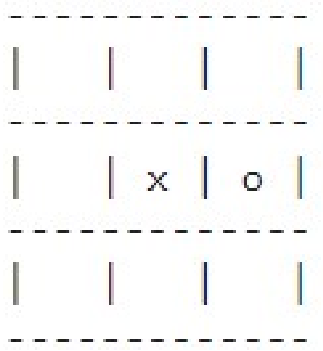


Figure 3: Initial Tic-Tac-Toe board.

Training the Agent

- The training of the reinforcement learning agent for the Tic-Tac-Toe game using the Q-learning algorithm.
- Two players, denoted as P1 and P2, initially play against each other and learn.
- Subsequently, the learned policy is loaded into the computer, and the agent plays against humans.
- During training, both P1 and P2 play 50,000 games, using their policies and simultaneously updating their policies.
- If P1 wins, it receives a reward of 1 and P2 gets a reward of -1.
- If P2 wins, it receives a reward of 1 and P1 gets a reward of -1.
- If the game ends in a draw, the reward is 0 for both P1 and P2.

Training the Agent

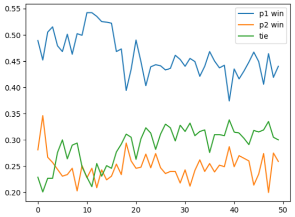


Figure 4: CP1CP2 - Both players P1 and P2 trained with constant epsilon. Win probabilities in each batch.

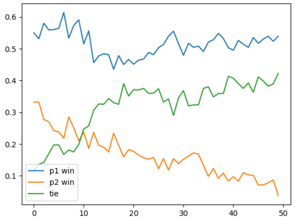


Figure 5: DP1CP2 - P1 trained with decay epsilon and P2 trained with constant epsilon. Win probabilities in each batch.

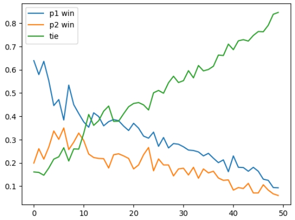


Figure 6: DP1DP2 - Both players P1 and P2 trained with decay epsilon. Win probabilities in each batch.

Testing the trained agent

- Both random players, P1 and P2, play against each other in a total of 50,000 games.
- The outcomes serve as a baseline to establish the expected normal performance.

Player	Winning Probability (%)
Random Player (P1)	57.40
Random Player (P2)	43.60

Table 3: Both players P1 and P2 are Random.

Testing the trained agent

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	CP1CP2	96.90	3.10	0.00
2	DP1CP2	98.40	0.00	1.60
3	DP1DP2	99.10	0.00	0.90

Table 4: Computer player (P1) plays against a Random player (P2).

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	CP1CP2	49.20	46.00	4.80
2	DP1CP2	49.50	46.40	4.10
3	DP1DP2	52.30	44.10	3.60

Table 5: Random player (P1) plays against a Computer player (P2).

Tic-Tac-Toe game using Neural Networks

- If a game has large number of states, storing Q values or State values in a dictionary/lookup table is not always feasible.
- Function approximation methods are used to estimate Q value or state value, using Neural networks.
- We defined the neural network architecture shown in Fig.7.
- This architecture allows the neural network to learn and approximate the state value associated with different states in the Tic-Tac-Toe game.

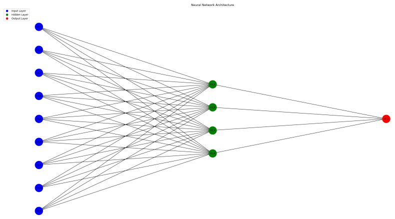


Figure 7: Neural network Architecture.

Training the Agent

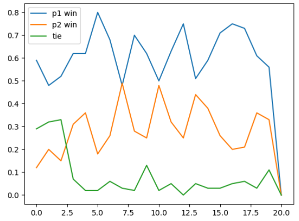


Figure 8: CP1CP2 - Both players P1 and P2 trained with constant epsilon. Win probabilities in each batch.

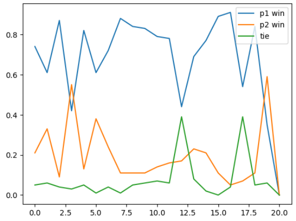


Figure 9: DP1CP2 - P1 trained with decay epsilon and P2 trained with constant epsilon. Win probabilities in each batch.

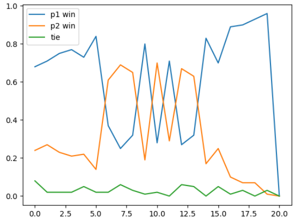


Figure 10: DP1DP2 - Both players P1 and P2 trained with decay epsilon. Win probabilities in each batch.

Testing the trained agent

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	CP1CP2	80.00	10.30	9.70
2	DP1CP2	80.20	12.40	7.40
3	DP1DP2	76.20	17.50	6.30

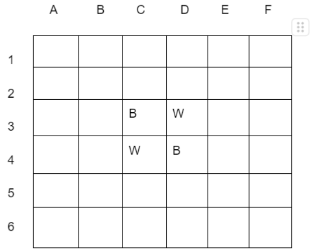
Table 6: Computer player (P1) plays against a Random player (P2).

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	CP1CP2	36.80	52.30	10.90
2	DP1CP2	40.10	52.20	7.70
3	DP1DP2	38.80	51.80	9.40

Table 7: Random player (P1) plays against a Computer player (P2).

Othello game using DQN algorithm

- Othello belongs to the family of two-player board games that were derived from Chess.
- It consists of a 6 x 6 board with black and white coins. Player 1 plays the black color coin denoted as 'B' and player 2 plays the white color coin 'W'.
- The game defines the eight possible directions to explore when searching for valid moves: up, down, right, left, and the four diagonals.



	A	B	C	D	E	F
1						
2						
3			B	W		
4			W	B		
5						
6						

Figure 11: Initial Othello board.

Training the Agent

- Each RL agent has this function approximation neural network. Two networks are employed, one for each player, where each network is responsible for predicting the state value for all possible actions.
- Here our input layer has 36 nodes, which corresponds to the number of cells in our game environment.
- Each network consists of an output layer that provides the state value.

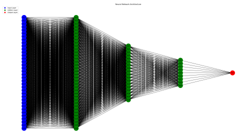


Figure 12: Neural network Architecture.

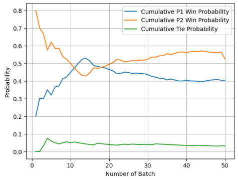


Figure 13: Cumulative probability where both players P1 and P2 trained with constant epsilon.

Testing the trained agent

- Both players P1 and P2 make random moves. This outcome serves as a baseline to establish the normally expected performance.

Player	Winning Probability (%)
Random Player (P1)	43.00
Random Player (P2)	52.00

Table 8: Both players P1 and P2 are Random.

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	CP1CP2	53.00	40.00	4.00
2	DP1CP2	73.00	24.00	3.00
3	DP1DP2	68.00	29.00	3.00

Table 9: Computer player (P1) plays against a Random player (P2) for 1000 games.

Testing the trained agent

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	CP1CP2	19.00	79.00	2.00
2	DP1CP2	25.00	72.00	3.00
3	DP1DP2	25.00	73.00	2.00

Table 10: Random player (P1) plays against a Computer player (P2) for 1000 games.

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	DP1DP2	23.00	76.00	1.00

Table 11: Random player (P1) plays against a Computer player (P2) for 3000 games.

Testing the trained agent

Test	Description	P1 win prob.(%)	P2 win prob.(%)	Tie prob.(%)
1	DP1DP2	72.00	23.00	5.00

Table 12: Computer player (P1) plays against a Random player (P2) for 3000 games.

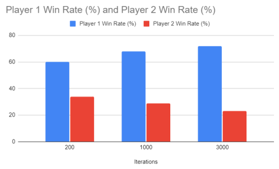


Figure 14: Computer player (P1) plays against a Random player (P2).

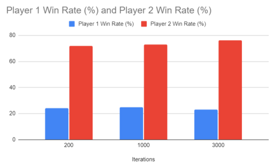


Figure 15: Random player (P1) plays against a Computer player (P2).

Conclusion

- We created an RL agent to learn the game of Tic-Tac-Toe using the Q-learning algorithm. The RL trained players, either player 1 (P1) or player 2 (P2), have an advantage compared to untrained/random players, and the probability of winning is higher compared to the baseline for P1 and P2.
- We then expanded our approach to Tic-Tac-Toe by implementing function approximation, particularly utilizing neural networks within TensorFlow's Keras.
- We addressed the complexity of Othello, a larger game with a state space surpassing typical machine memory capacities. We observed that the learned player, whether P1 or P2, had a higher chance of winning because reinforcement learning assists players in learning strategies that enhance their performance.
- These findings indicate that Reinforcement Learning helps players learn strategies that improve performance.

Future Work

- In our future work, we will apply RL to even larger games, such as Chess, with an 8x8 board and a state space of 10^{44}.
- We will explore deep reinforcement learning techniques like SARSA and PPO etc. to address the complexities of these board games, thereby advancing our understanding and capabilities in strategic decision-making environments.
- During the training process of RL agents, we will leverage useful techniques such as Monte Carlo Tree Search (MCTS) and human-expert knowledge to enhance the agent's performance.
- Evaluating the performance of RL agents in Chess will require sophisticated metrics beyond simple win-loss ratios.

[1] Guru99. *Reinforcement Learning Tutorial*. 2024. URL: https://www.guru99.com/reinforcement-learning-tutorial.html.

[2] Ugur Suay and Claudio Gonzalez. "Reinforcement Learning Explained Visually: Part 4 — Q-Learning Step-by-Step". In: *Towards Data Science* (2022). URL: https://towardsdatascience.com/reinforcement-learning-explained-visually-part-4-q-learning-step-by-step-b65efb731d3e.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

THANK YOU