Reinforcement Learning using Q-learning Tic Tac Toe Game

Shamitha Thumma (sxt200080), Swetha Malaivaiyavur Elayavalli (sxm220052),
Rini Patel (rxp220124), and Tinsy John (tej180001)

I. ABSTRACT

There are four basic types of learning in Machine Learning, namely supervised learning, unsupervised learning, semi-supervised learning, and Reinforcement learning. As we start from supervised learning and move towards the other types, we can observe that the amount of data provided for the learning reduces - features along with their associated classes are provided for supervised learning;, data features without class labels are provided to the unsupervised algorithm which has to combine similar data together to classify incoming test data; data both with labels and without labels provided to semi supervised learning algorithms and finally, model space along with rewards, paths and final goal provided to reinforcement learning. The error metrics vary in each type of learning, from knowing where exactly the mistake was, measure of how wrong the algorithm predicted to just knowing if the end result was right or wrong. In the final type of learning, the algorithm does most of the work to find the correct path based on rewards for every action. Rewards could be positive as well as negative based on what the environment is that the learning agent is facing. Positive reinforcement is used in cases where the problem requires the learner to maximize performance i.e. the action has to be done more and negative reinforcement is used when a particular behavior has to be strengthened i.e. a particular action has to be avoided in the environment. For this project we would be focussing on the former type of reward. In this tic-tac-toe game using Q-learning, which is a type of reinforcement learning, there is no model that guides the Q-learning agent throughout the course of its actions. Through multiple trials or game play, the agent refines its strategy, adapting to the limited data scenario. The code examines various hyperparameter combinations, including training iterations, learning rate, decay rate, and exploration-exploitation thresholds, unveiling insights into the agent's learning dynamics. Training results showcase the agent's proficiency in winning, losing, or drawing under diverse conditions. Additionally, the code offers an interactive interface for a human player to engage with the trained Q-learning agent, providing a hands-on experience and illustrating the adaptability of reinforcement learning in scenarios with constrained data. This work contributes to understanding how Q-learning can effectively navigate strategic decision tasks with limited training data. This project elaborates how this Q-learning model is applied to learn how to win the tic-tac-toe game based on just the reward for if it won the game or not.

II. INTRODUCTION & BACKGROUND WORK

Machine learning is a subset of artificial intelligence that focuses on using data and algorithms to perform tasks that emulate human thinking and learning abilities. [1] Its fundamental goal is to enable computers to learn from data and make informed predictions or decisions based on it. It differs from traditional programming, where we, as developers, had to explicitly program the computer to perform a task. Machine learning consists of three main types of learning: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves using labeled data to train the algorithm. Unsupervised learning entails giving the algorithm unlabeled data to train it and is tasked to find patterns within the data. Lastly, reinforcement learning, the focus of this project, is about training an agent to make decisions to achieve a specific goal. It employs the notion of rewards and penalties, aiming to get the agent to maximize the reward in its environment.

Reinforcement learning enables a machine to address complex problems iteratively through trial and error. The features of reinforcement learning make it a suitable approach for our Tic Tac Toe game. Our specific goal was to use a Q-learning agent for our Tic Tac Toe game. Q-learning is a model-free reinforcement learning where the agent iteratively learns by exploring the environment and updating its model. [3] Its key components include agents that operate within states, taking actions that yield rewards, and Q-values representing the metrics for measuring actions in specific states throughout episodes. It focuses on making an optimal action based on the current environment state. [2]

III. THEORETICAL & CONCEPTUAL STUDY

Reinforcement Learning is a machine learning approach where an agent learns by interacting with an environment. There is an environment, goal, several paths, obstacles which the learning agent will interact with. The environment is the model space which contains all parameters- obstacles i.e, points which the agent would have to avoid in order to maximize rewards and reach the goal efficiently; the paths which present various options for the agent to pass through and the rewards or punishments which are provided to the agent to let it know how it performed. Sometimes, rewards are provided on a short term and long term basis, the algorithm has to optimize its rewards based on those factors even if it means getting lots of negative rewards until it receives a higher reward.

As can be seen in Fig 1, the agent takes actions towards reaching the goal, receives feedback in the form of rewards or penalties, and aims to learn a strategy that maximizes cumulative rewards over time [4]. It involves exploration of actions, exploitation of learned strategies, and finding an optimal decision-making policy. Popular algorithms include Q-learning, and RL is applied in diverse fields such as robotics, gaming, and finance [6].

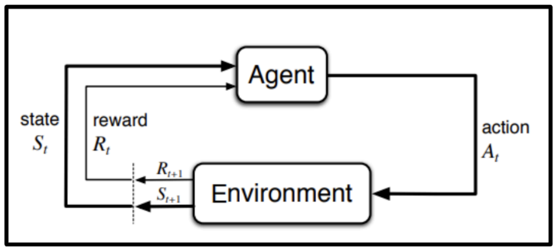

Fig. 1

There are several types of learning in reinforcement learning Q-learning is a reinforcement learning algorithm designed for agents to learn decision-making in an environment. It empowers agents to make optimal decisions by assigning Q-values to state-action pairs, representing expected cumulative rewards. 'Q' represents the quality of an action in a given state, and the algorithm helps the agent acquire an optimal strategy by assigning Q-values to state-action pairs. These Q-values represent the expected cumulative reward for taking a specific action in a particular state. Q-learning involves updating a Q-table iteratively. Parameters including learning rate, decay rate, and the explore-exploit threshold also influence the learning process. The algorithm in each iteration determines whether to explore a new action or exploit its already learned experiences to maximize its overall reward at the end [5]. The formula for the updated Q-value after every action is as follows:

$$New\_Q\_Value = Current\_Q\_Value + Learn\_Rate * (Reward + Decay\_Rate * Best\_Next\_Move - Current\_Q\_Value)$$

The learn rate is what the agent uses to determine how much to update the Q-value by in each iteration for an action it takes. The decay rate is the weight of the future reward for the agent and it is used so the agent can make a choice between taking an action that will either give it an immediate reward or an action that will guarantee a better future reward. The explore-exploit threshold is used by the agent to pick whether to make a random next move or use its previously learned experiences to pick the best next move.

Tic Tac Toe is a two-player game that is played on a 3x3 grid. Players take turns filling empty cells with either an X or O (which denotes what player it is). To win the game, a player has to get a sequence of three of their symbols horizontally, vertically, or diagonally. The game ends when one player wins, or the grid is completely filled without a row of a player's symbols, meaning that the game ended in a draw. In reinforcement learning using Q-learning, Tic Tac Toe can be represented as distinct game board states each of which is a unique environment. An action for the agent would be to place either an X or O on the board (depending on the symbol assigned to the agent), leading to rewards for winning or ending the game in a tie. Q-values, representing anticipated rewards, guide the agent's decisions [7]. If the agent wins a game, the reward given is 1, for the loss the reward is -1, and the reward for a tie is 0.8. We propagated a relatively high reward for tie because we wanted to incentivize the agent to tie a game rather than lose a game and thus reward the agent for producing a tie.

When the agent chooses a possible move on the 3x3 board, the move is represented by an (a,b) tuple where a is the row and b is the column. The current game state is the flattened values of the 3x3 board that are filled with either O, X, or nothing based on the move that was made. The Q-table for the agent is formed as the agent learns the game and makes decisions based on the reward that is propagated for each game state. The agent calculates potential reward and makes a decision against immediate reward or future reward. The agent then picks the Q-value that will help it maximize the potential of its next move in the game. The next move the agent picks is determined in the pick_next_action where the agent compares its move against the explore-exploit threshold to determine whether it is better for the agent to make a random move or instead use its previous learning experience. We generated a random number between 0 and 1. If the number was less than the threshold, the agent would pick a random next move out of the possible moves for the current game state. If the number generated was greater than the threshold, the agent would make a next move based on its previous actions.

The agent strikes a balance between exploring new moves and exploiting known strategies. Through repeated gameplay iterations, the agent refines its strategy to learn a policy that maximizes cumulative rewards over time. The ultimate objective is for the agent to develop a decision-making approach that ensures the highest total reward.

IV. RESULTS & ANALYSIS

Our Tic Tac Toe game has two players: agent 1 and agent 2. Agent 1 is the Q-learning agent and Agent 2 is the agent that represents a random player. The Agent 1 Win % represents the number of times that the Q-learning agent won and the Agent 2 Win % represents how many times the random player won. The Tie % represents how many games led to a tie. The learn rate is the learning rate of the Q-learning algorithm for Agent 1 after each round, or how much we update the Q-value in each step. The decay rate is the value of the future reward for the Q-learning agent. This is used because the agent needs to evaluate and make a choice between the value of an action that will give an immediate reward versus a future reward. The explore-exploit threshold is the threshold which dictates that based on the random number generated between 0 and 1, if the number is less than the threshold, the agent will make a random move but if it is greater than the threshold, the agent will rely on its previous learning experiences to make a move.

Iterations:	**Learn Rate:**	**Decay Rate:**	**Explore-Exploit Threshold:**	**Agent 1 Win %:**	**Agent 2 Win %:**	**Tie %:**
1000	0.1	0.8	0.2	0.73	0.15	0.12
			0.3	0.66	0.29	0.05
		0.7	0.2	0.68	0.25	0.07
			0.3	0.69	0.2	0.11
	0.2	0.8	0.2	0.67	0.16	0.17
			0.3	0.63	0.25	0.12
		0.7	0.2	0.68	0.27	0.05

			0.3	0.56	0.35	0.09
10000	0.1	0.8	0.2	0.74	0.19	0.07
			0.3	0.63	0.24	0.13
		0.7	0.2	0.53	0.39	0.08
			0.3	0.6	0.23	0.17
	0.2	0.8	0.2	0.65	0.29	0.06
			0.3	0.55	0.33	0.12
		0.7	0.2	0.7	0.21	0.09
			0.3	0.53	0.36	0.11

Fig 2. Logs with Varying Parameters

Iterations:	**Learn Rate:**	**Decay Rate:**	**Explore-Exploit Threshold:**	**Agent 1 Win %:**	**Agent 2 Win %:**	**Tie %:**
1000	0.1	0.8	0.2	0.65	0.26	0.09
2000	0.1	0.8	0.2	0.69	0.21	0.1
5000	0.1	0.8	0.2	0.72	0.21	0.07
10000	0.1	0.8	0.2	0.77	0.2	0.03
15000	0.1	0.8	0.2	0.72	0.21	0.07
20000	0.1	0.8	0.2	0.65	0.26	0.09

Fig 3. Logs with Fixed Parameters & Varying Iterations

We tested our Q-learning model using Agent 1 which was the QLearning agent against Agent 2 which represents a random player. Unlike Agent 1 which uses Q-learning, Agent 2 just picks a random move out of the available possible moves for a given stage of a Tic Tac Toe game. Agent 1 makes a move based on the Q-learning algorithm after evaluating possible rewards and looking at the q-values in its Q-table. As shown in Fig 2, Agent 1 (the model) tended to perform better most of the time when the learning rate was lower (0.1), when the decay rate was higher (0.8), and when the explore-exploit threshold was lower (0.2). When the decay rate is high, according to the Q-learning formula, a higher reward is propagated to Agent 1 and so the agent may have made better moves for the higher decay rate. Since the explore-exploit threshold was lower, the agent may have used its previous learning experiences more of the time rather than making a random choice, allowing Agent 1 to perform better against Agent 2 for the lower explore-exploit threshold value. The Tie % generally stayed consistent but it was higher for the 10000 iteration instances than the 1000 iteration instances. This may have been due to the fact that we still propagated a high reward value (0.8) for ties, which may have incentivized the agent to tie in more scenarios rather than lose as it learned on more instances.

Based on Fig 2, the best values we tended to receive for Agent 1 was with learn rate = 0.1, decay rate = 0.8, and explore-exploit threshold = 0.2. Thus, we kept these same values when testing Agent 1 across multiple different values of iterations. As can be seen in Fig 3, Agent 1 did the best for 10000 iterations but then the Agent 1 Win % started to go down. Raising the number of iterations increased Agent 1 Win % up until after 10000 iterations whereby then it stayed relatively constant.

V. CONCLUSION & FUTURE WORK

In conclusion, this project successfully implemented a reinforcement learning approach, specifically Q-learning, to train an agent for playing Tic-Tac-Toe. The results indicate that the Q-learning agent, represented by Agent 1, demonstrated proficiency in making strategic moves against a random player (Agent 2). The choice of hyperparameters, including learning rate, decay rate, and explore-exploit threshold, significantly influenced the agent's performance. Optimal values, such as a lower learning rate, higher decay rate, and lower explore-exploit threshold, led to superior outcomes.

The iterative testing across different numbers of training iterations showcased the agent's learning dynamics, with the best performance observed around 10,000 iterations.

For future work, further exploration could involve experimenting with alternative reinforcement learning algorithms, extending the Q-learning model to more complex games, and enhancing the interactive interface for a more engaging user experience. Additionally, the impact of additional features and complexities in the environment could be studied to assess the adaptability of the Q-learning agent. Overall, this project provides a foundation for continued research in applying reinforcement learning techniques to strategic decision-making scenarios with limited training data.

VI. REFERENCES

[1] “What is machine learning?,” IBM, https://www.ibm.com/topics/machine-learning (accessed Dec. 5, 2023).

[2] M. Banoula, “What is Q-learning: Everything you need to know: Simplilearn,” Simplilearn.com, https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning (accessed Dec. 5, 2023).

[3] N. Singh, “Understanding Q-learning: A powerful reinforcement learning technique,” Medium, https://medium.com/@navneetsingh_95791/understanding-q-learning-a-powerful-reinforcement-learning-technique-29a3da36f611 (accessed Dec. 5, 2023).

[4] View of Reinforcement Learning: Playing Tic-tac-toe, https://www.jsr.org/index.php/path/article/view/1739/1182 (accessed Dec. 5, 2023).

[5] S. Bhatt, “Reinforcement learning 101,” Medium, https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292 (accessed Dec. 5, 2023).

[6] A. Violante, “Simple reinforcement learning: Q-learning,” Medium, https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56 (accessed Dec. 5, 2023).

[7] C. Friedrich, “Part 3 - tabular Q learning, a tic tac toe player that gets better and better,” Medium, https://medium.com/@carsten.friedrich/part-3-tabular-q-learning-a-tic-tac-toe-player-that-gets-better-and-better-fa4da4b0892a (accessed Dec. 5, 2023).