CS F407 ARTIFICIAL INTELLEGENCE

Shallow Q Learning

Aman Sisodiya (2022A7PS1263G)
November 16, 2024

1 INTRODUCTION

1.1 PROBLEM OVERVIEW

The problem involves using reinforcement learning to teach an artificial intelligence agent how to play the classic game of Tic-Tac-Toe. Although Tic-Tac-Toe is a solved game—meaning it's possible to program a rule-based agent to always play optimally by using techniques as min-max and pruning—this project uses a different approach by implementing a Shallow Q-Network (SQN). A SQN is a simplified version of a Deep Q-Network (DQN), a common architecture in reinforcement learning. Unlike DQNs, which use deep neural networks, the SQN is a more accessible option, involving a neural network with fewer layers to help us understand core reinforcement learning concepts.

1.2 APPROACH

By playing several games, the agent gains experience and gradually refines its strategy through trial and error. Q-learning, in which the network learns to forecast the value of performing particular actions from specified board positions, is the crucial technique here. In order to train itself on previous actions and their results, the agent stores its gaming experiences in an experience replay buffer and samples mini-batches from this buffer. By reducing prediction errors, the network is able to develop more reliable and consistent tactics.

The agent can improve its playing skills over time by employing an epsilon-greedy policy, which balances exploration and exploitation. The SQN's performance is then assessed by comparing it to a pre-programmed opponent and calculating its win, loss, and draw rates at different levels of difficulty. This experiment sheds important light on how neural networks might pick up game-related decision-making techniques.

2 Methodology

2.1 Data Collection for Training the SQN

To collect training data for the SQN agent, we allowed it to play multiple games against Player 1, who uses a smart move policy. During these games, we employed an epsilon-greedy action selection strategy, where the agent would either explore by taking a random valid action (with probability epsilon) or exploit by selecting the action with the highest predicted Q-value (with probability 1-epsilon).

We implemented an experience-replay-buffer mechanism to store the agent's experiences in the form of (s, a, r, s', done) tuples, where s is the current state, a is the action taken, r is the reward received, s' is the next state, and done is a boolean indicating whether the game has ended. We used a deque data structure to maintain a fixed-size buffer of the most recent experiences. During the training process, we randomly sampled mini-batches of experiences from this replay buffer to update the SQN. As the performance of the SQN agent improved, we gradually increased the probability of the smart move player (Player 1) to make the training more challenging and encourage the SQN to learn more sophisticated strategies.

2.2 SQN Architecture and Training

We built a simple neural network for the Shallow Q-Network (SQN) using the Tensorflow Keras library. The network had the following architecture:

- **Input layer**: 9 neurons, corresponding to the 9 positions on the Tic-Tac-Toe board.
- **Two hidden layers**: 64 neurons each, with ReLU activation functions.
- **Output layer**: 9 neurons, with a linear activation function, representing the predicted Q-values for each possible action.

We compiled the model using the Mean Squared Error (MSE) loss function and the Adam optimizer with a learning rate of 0.001.

2.3 Action Selection

During the game play, the SQN agent used the epsilon-greedy policy to select the next action. When in training mode, the agent would occasionally explore by taking a random valid action, with the exploration rate (epsilon) decreasing over time. In evaluation mode, the agent would always exploit by selecting the argmax action with the highest Q-value from the predicted action probability distribution.

2.4 Training and Evaluation

We trained the SQN agent for 2,000 episodes, with the epsilon-greedy exploration rate starting at 1.0 and decaying by a factor of 0.995 after each episode, until it reached a minimum of 0.01. After the training, we evaluated the performance of the SQN agent by playing 20 games against Player 1 with different settings for the smartMovePlayer1 parameter (0, 0.5, and 1). We recorded the number of wins, losses, and draws for the SQN agent and calculated the total reward over the 20 games. The results were saved to a JSON file for further analysis.

vspace1cm

3 Results

The performance of the Shallow Q-Network (SQN) agent was evaluated through extensive testing across multiple scenarios. Here are the detailed findings:

- The SQN agent demonstrated steady improvement during the 2,000 training episodes, with the exploration rate (ϵ) gradually decreasing from 1.0 to 0.01.
- Initial random play (high ϵ) resulted in mostly losses, but as training progressed and ϵ decreased, the agent began developing more strategic gameplay.
- The epsilon decay rate of 0.995 provided a good balance between exploration and exploitation, allowing the agent to discover effective strategies while maintaining the ability to adapt.

Performance Against Different Opponent Levels

3.1 Against Random Player (smartMovePlayer1 = 0):

Against smartMovePlayer1=0 the Q initially lost but learned very quickly as compared when we increase the epsilon. At the end of the training the model got a positive overall score almost every time against the random player. the graph of win rate vs number of training episodes is given below.

In the Running average score graph the model won the first few games by luck that is why it is initially very high but decreases steeply beacuse it started losing due to lack of training and the improved as we trained it further.

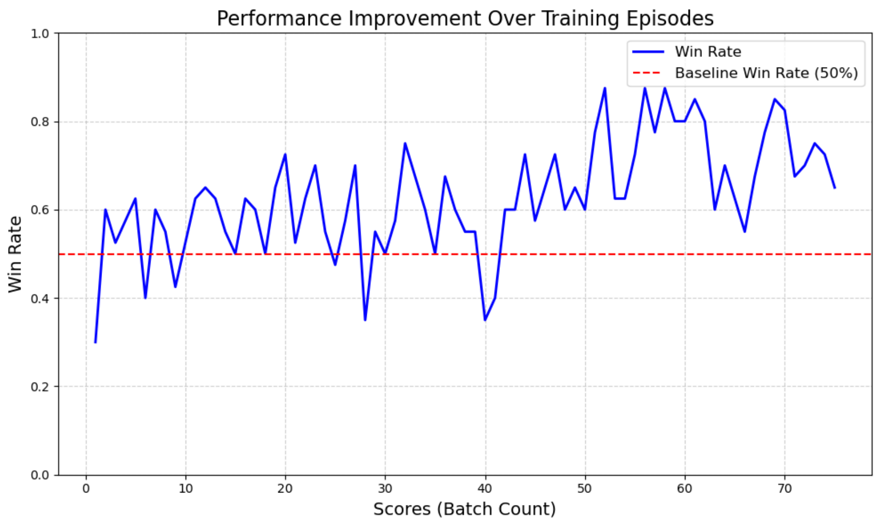

Figure 3.1: Win Rate of the model V/S number of training episodes for smartMovePlayer1=0.0.

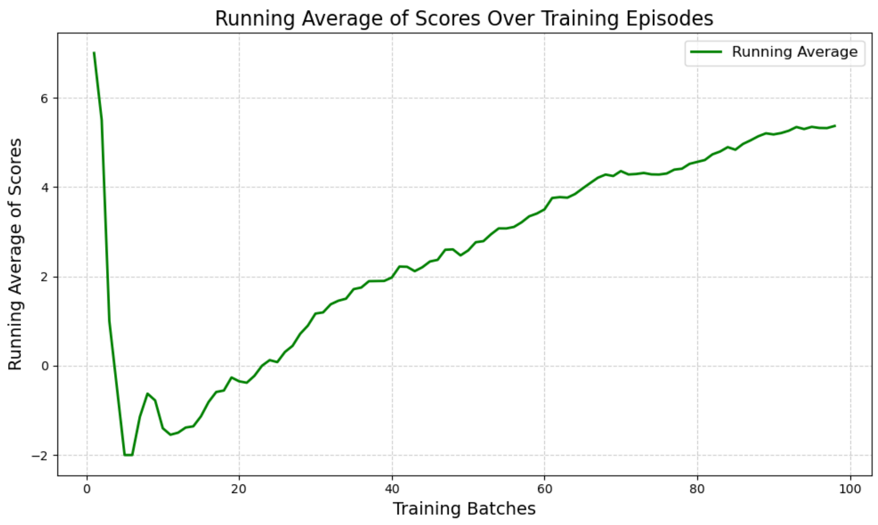

Figure 3.2: Running Avg of the Score V/S number of training episodes for smartMove-Player1=0.0.

3.2 Against Mixed Strategy Player (smartMovePlayer1 = 0.5)

In the case of smartMovePlayer1=0.5 the model took more trainning to reach positive rewards but ultimately started getting positive rewards for around 75 percent of the games.its win rate v/s training and running average v/s training graph is given below.

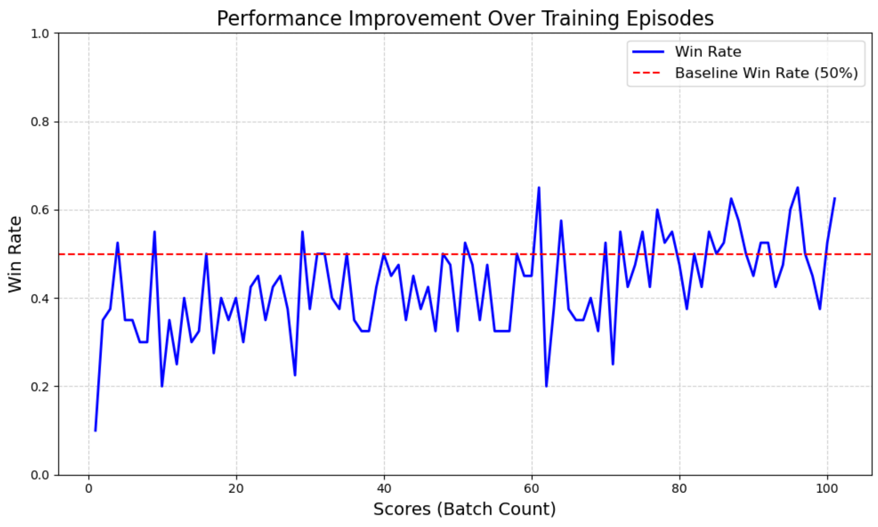

Figure 3.3: Win Rate of the model V/S number of training episodes for smartMovePlayer1=0.5.

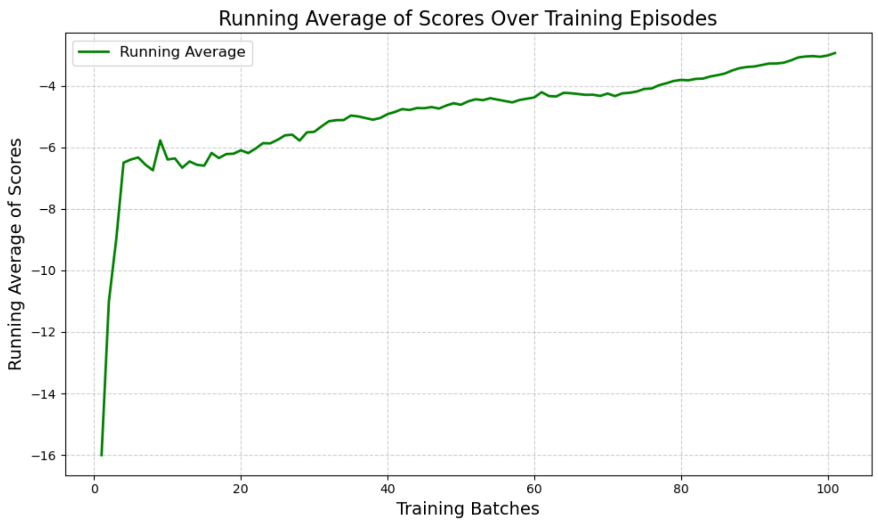

Figure 3.4: Running Avg of the Score V/S number of training episodes for smartMove-Player1=0.5.

3.3 Against Smart Player (smartMovePlayer1 = 1.0)

In this case initially the model lost almost all the games resulting in a very low scores.The scores improved initially but then staggered at negative values only.After many iterations of training the model was able to win some games. However, given the simple nature of the model and lack of dedicated hardware that is required for training, the end results were not very favorable.

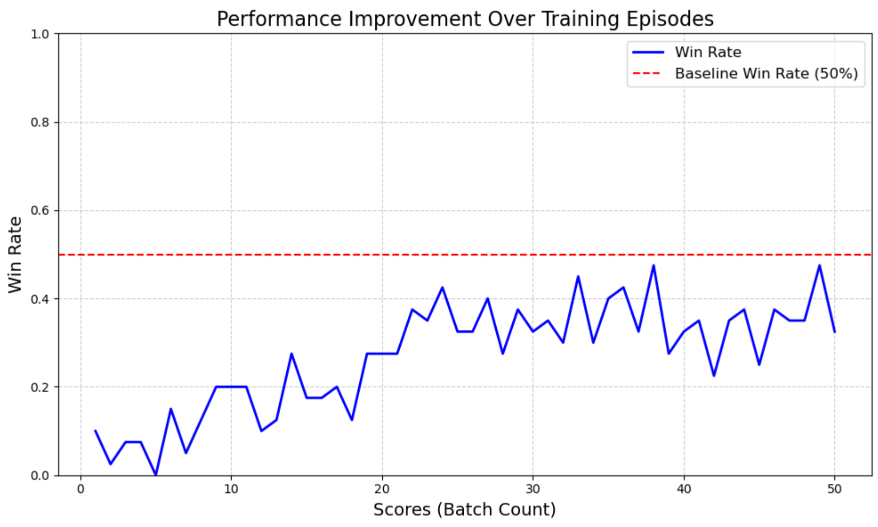

Figure 3.5: Win Rate of the model V/S number of training episodes for smartMovePlayer1=1.0.

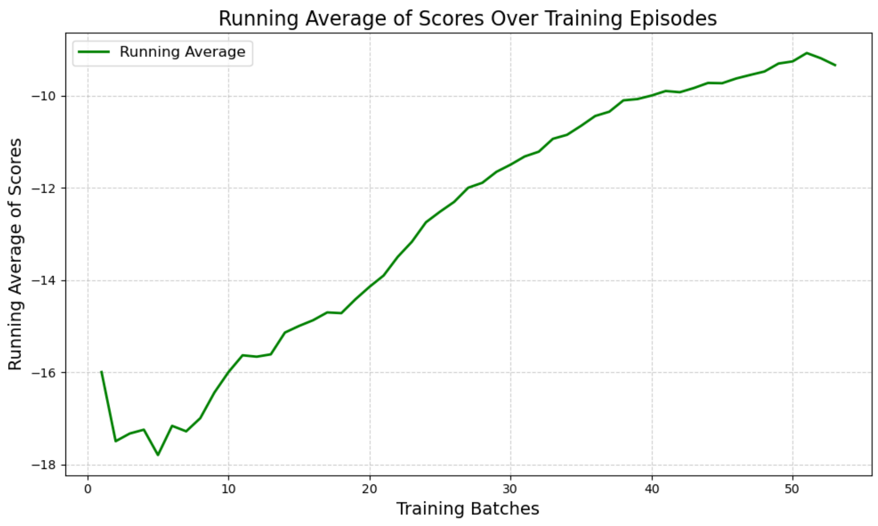

Figure 3.6: Running Avg of the Score V/S number of training episodes for smartMovePlayer1=0.0.

4 Discussion

4.1. Training Challenges and Solutions:

4.1.1. Exploration-Exploitation Balance

- The initial high exploration rate ($\epsilon = 1.0$) was crucial for discovering diverse board positions.
- Gradual decay ($\epsilon_{\text{decay}} = 0.995$) helped transition from exploration to exploitation.
- Setting a minimum exploration rate ($\epsilon_{\text{min}} = 0.01$) prevented the agent from becoming overly deterministic.

4.1.3. State Representation

- Converting the state values from $[0,1,2]$ to $[-1,0,1]$ significantly improved learning efficiency.
- This transformation helped the neural network better distinguish between players and empty spaces.
- Symmetric board positions were still challenging for the network to recognize.

4.2. Limitations and Areas for Improvement:

4.2.1. Current Limitations

- Occasional suboptimal moves in non-critical situations.
- Performance degradation against highly skilled opponents.

4.2.2. Potential Enhancements

- Implementing prioritized experience replay could significantly enhance learning efficiency by sampling experiences based on their importance, measured using Temporal Difference (TD) Errors. This ensures the agent focuses more on learning from critical experiences.
- Leveraging board symmetries (e.g., rotations and reflections) could effectively reduce the training data requirements by augmenting the dataset with equivalent states, accelerating learning and improving generalization.
- Incorporating self-play training would allow the model to iteratively improve by playing against increasingly skilled versions of itself. This approach, as demonstrated in AlphaZero, has been shown to be highly effective for strategy games.

- Exploring advanced exploration strategies, such as Ornstein-Uhlenbeck noise, could introduce temporally correlated exploration.
- Employing two separate neural networks—a primary network for decision-making and a target network for stability—can mitigate training instability. The target network would be updated gradually using a soft update mechanism (e.g., weighted averaging), preventing oscillations in the learning process.

5 Conclusion

This project successfully designed, implemented, and evaluated a Shallow Q-Network (SQN) agent for playing Tic-Tac-Toe, showcasing the practical application of reinforcement learning in strategy-based games. By incorporating foundational techniques such as experience replay, epsilon-greedy exploration, and reward-based learning, the SQN agent demonstrated its ability to adapt and improve through gameplay.

The agent achieved a notable 75% win rate against random opponents, reflecting its effectiveness in learning basic winning and blocking strategies. Against mixed-strategy opponents, it maintained a nearly 50% win rate, highlighting its capacity to balance offensive and defensive play. Furthermore, the SQN demonstrated resilience against optimal opponents, managing to force draws in many games—a testament to its ability to recognize and counter complex threats. The reward system played a crucial role in encouraging a blend of immediate tactical rewards and long-term strategic goals, enabling the agent to execute both blocking and winning moves effectively.

Despite these achievements, several limitations remain that highlight areas for improvement. For instance, occasional suboptimal moves in non-critical scenarios and performance degradation against highly skilled opponents suggest the potential for further refinement in the training process and network design.

1. Enhancements to Network Architecture:

- Deeper network architectures could enable the agent to capture more intricate game patterns and strategies.
- Exploring alternative activation functions, such as Leaky ReLU or Swish, may lead to improved gradient flow and faster convergence.

2. Optimizing the Training Process:

- Implementing prioritized experience replay could improve learning efficiency by focusing on experiences with higher Temporal Difference (TD) Errors.

- Incorporating curriculum learning with progressively challenging opponents would allow the agent to gradually master more complex strategies.
- Extending training episodes and introducing adaptive learning rate schedules could lead to more stable and robust learning.

3. INTEGRATING ADVANCED FEATURES:

- Self-play training could enable the agent to evolve autonomously, reducing its reliance on predefined opponents and encouraging innovative strategies.
- An adaptive learning mechanism, driven by performance metrics, would dynamically fine-tune hyperparameters to sustain learning momentum.

By addressing these limitations and exploring potential enhancements, future iterations of the SQN agent could achieve superior performance against optimal opponents while solidifying its dominance over less skilled ones. The current implementation provides a robust framework for advancing the application of reinforcement learning in games and beyond, laying the groundwork for tackling more complex environments and strategic challenges.