Report on Shallow Q-Network (SQN) for Tic-Tac-Toe

In spite of being straightforward understanding game, its given an opportunity to design strategies against artificial intelligence. Here in this project, the player 2 adopts a Shallow Q-Network (SQN) agent and aims to win against the player 1. An SQN agent is trained using reinforcement learning principles by playing the game and optimizing its performance by learning from the rewards that it gets for its actions in the game.

And we also have a learning process that makes use of:

- Q-Learning: An algorithm that is based on Bellman equations for future reward predictions.

- Experience Replay: A method which involves the use of past experience to help reinforce the learning process.

- Epsilon-Greedy Exploration: A strategy that helps to balance between testing new actions and using already acquired knowledge.

The purpose of incorporating a neural network is to allow the SQN agent to seek to learn the optimal Q-function as a guide to the best course of action for any state of the game.

2. Methodology

2.1 Game Environment

The environment of the tic-tac-toe contains a 3*3 grid and incorporates the following elements.

- States – a 9-element array – 0 means empty cell, 1 means Player 1's move while 2 means Player 2's move

- Actions – indexes of available cells to make a move

- Rewards:

  - +1 for a win.

  - 0 for draw.

  - -1 for lost.

  - +0.1 for getting closer to winning.

- Player 1 behavior: Random motion or strategic performed with some probability.

2.2: Shallow Q-Network (SQN) Design

The Shallow Q-Network (SQN) design uses Tensorflow to model Function approximation with deep networks for solving the optimal Q-function.

- Input Layer: Utilizes the extended 9 coat readable board's state as input.

- Hidden Layers: Two dense, 128 neurons with ReLU activation layers are used to help understand the game states.

- Output Layer: Predicts Q-values for all 9 available actions, performing the expected reward for each move.

The network learns to estimate the Q-values for each action in a given state. Its objective is to approximate the optimal Q-function, which satisfies the Bellman equation:

$Q(s,a)= r + \gamma \cdot \max Q(s',a')$

Here:

- Q(s,a) represents the current state's action-value.
- r is the immediate reward.
- γ=0.95 is the discount factor for future rewards.
- s' and a' are the next state and action, respectively.

The training of the network is based on the process of minimizing the error between the estimated Q-values and the derived Q-values from the Bellman equation. This loss function is optimized via stochastic gradient descent and backpropagation to enhance and alter the weights of the network in stages. The SQN also uses the Bellman equation for value functions, but instead of storing value functions in a look-up Q-table, the network itself operates to learn the optimal value function or Q-function. This continues until there is a sufficiently low loss value and the network is able to create a close approximation of the optimal Q function. 2.3 Exploration-Exploitation Policy An epsilon-greedy strategy is used to determine the moves of the agent where

- Starts with epsilon = 1.0 to encourage exploration.

- Epsilon is decayed at a rate of 0.995 to promote exploitation instead.

- On reaching the value of epsilon equaling 0.05, decay stops – this allows for slight exploration.

2.4 Training Process

The training consists of:

1. Game Simulation: The agent plays out 100 games with Player 1.

2. Replay Buffer: Collects transitions (s,a,r,s′,done) for a maximum of 5000 for each buffer.

3. Mini-Batch Training: Takes 64 transitions from the buffer and computes the target Q values using the Bellman equation and adjusts the network weights.

4. Epsilon Decay: Lessens the value of epsilon after every episode in order to change the focus from exploration to exploitation. 2.

5 Evaluation After the training is complete, the agent is put to the test by playing 10 games against a random player. The following criteria are used to evaluate the performance:

• Win Rate: Number of games won by the agent.

• Loss Rate: Number of games won by the opponent.

• Draw Rate: Number of games that have no winner.

---

3. Results
Epsilon Decay:
The epsilon value, determining the balance between exploration (trying out new actions) and exploitation (relying on learned strategies), starting with an initial value of 1.0 (implying that the agent explored all possible moves randomly), was decayed over the course into an approximated value of 0.05. This set-up allowed the agent to explore widely during the early stages and develop a very wide first experience. As the value of epsilon decayed, the agent became increasingly reliant on the strategies it had learned, rather than on random exploration. This concomitant decay is essential for the agent's success, as it would avert the risk of the agent being stuck inside its suboptimal strategies while allowing it to focus on exploiting the best-known strategies as training progressed.
Reward Trend:
The reward trend over the episodes describes how well the agent is learning. At the start, rewards varied significantly, since the agent did not yet possess an intuitive sense for how the game worked. Over the course of the training process, the average rewards steadily increased, reflecting the agent's deeper understanding, leading to better moves and eventually winning more games. By the end of the training, rewards for the agent had largely increased, which demonstrated that the agent had learned effective decision-making and strategy improvement in terms of blocking the opponent and setting up winning combinations.

3.2 Evaluation Results

Ten games were held on the training process, thus marking an evaluation on the performance of the agent against a random opponent. The game results were:

- Wins: 8
- Losses: 1
- Draws: 1

Such environments served as proof that the agent could implement the strategies learned during training into competitive use. Eight wins out of ten games showed evidence that the agent had convincingly learned how to play Tic-Tac-Toe and took optimal moves. Given such conditions, it was likely that the opponent's strategy or randomness in moves led the agent into situations where its strategy could not succeed. The draw implies that, although the agent was efficient enough to have a chance to win, there were also cases when both players

made optimal moves, entering into a stalemate.

3.3 Graph Analysis

The following graphs were generated during the evaluation process to delineate the agent's learning progress.

Epsilon Decay vs. Episodes:

The graph represents the gradual decay of epsilon over the course of training. It visually represents how the agent shifted from an exploration-heavy approach to one more reliant on exploitation. With an initial epsilon of 1.0, the agent explored completely until the reference rounds, but as the episodes proceeded, epsilon decayed to 0.05, showing that the agent increasingly favored exploitation strategies, having applied the proper learned technique for another episode. Therefore, this transition is quite critical, as it ensures that the agent explores the environment effectively before it settles on its best-known strategies.

Win Rate vs. Episodes:

This graph traces the win rate throughout the training episodes. As well expected, the agent's win rate increased effectively, especially after the 50th episode. This is indicative of the fact that the agent came to internalize effective strategies, such as blocking the winning moves of the opponent and placing its own marks at strategic locations on the board. The rise in the win rate beyond this level denotes that the agent had become more capable of winning, in essence, indicating that its learning algorithm was beginning to take hold with an effective shift from exploration into exploitation.

Average Reward vs. Episodes:

The average reward plot demonstrates how the collective performance of the agent, as such, improved over time. Initially, these rewards were sporadic, being indicative of the agent's trial-and-error learning process through which it came to grips with the rules and tactics of the game. However, with the progress of time, the average reward increased corresponding to the success of the agent due in part to increasingly advantageous decision-making. The upward track throughout time corresponds to a more systematic decision-making ability by the agent which indicates that it learned ways in which to maximize its rewards through play likely tailored for success.

.

---

4. Discussion

Challenges

1. Limited Rewards:

   One of the primary difficulties experienced in the course of training is predominantly the limited reward system. In the game of Tic Tac Toe rewards are provided solely at the end of the game whether the game is a win, a loss or a draw. This implies that for several steps in the course of the game, the agent does not get any explicit indication of the quality of its actions. This feedback latency can prove challenging to the agent when it comes to learning what moves, in particularly, caused the end result, especially when the agent is still learning the ropes. As a consequence, the agent was heavily dependent on its experience replay mechanism, which was storing all previous sessions and practising with them to perfect the strategies employed. The restless nature of the learning and limited feedback per episode meant the agent was forced to adopt different plans and utilize the effects of its actions over time rather than rewards after performing an action.

2. Exploration-Exploitation Balance:

   In reinforcement learning, it is very important to find a good tradeoff between exploration (selecting actions that have not been tried yet) and exploitation (selecting actions that have been learnt and known to be effective). The epsilon greedy policy is intended to alleviate this problem whereby the agent is allowed to take wide ranging actions at first, and as training goes on, the agent will focus on the learned action only. However, this was a sensitive issue because of the decay of epsilon. If epsilon was reduced too fast, the agent could lose interest in seeking new options and stop trying out different strategies that may have been useful for the agent. Conversely, if the development of epsilon was slow, the agent would continue searching through the options which were not required because it would make the learning process ineffective. Therefore, appropriate orientation by providing considerable time to the two mechanisms is paramount for effective learning. The decrease in the value of the epsilon parameter from its initial value of 1.0 to about 0.05 within the specified periods enabled the agent to efficiently shift its focus from exploration to exploitation while still allowing some room for exploration at the latter training periods.

3. Computation Overhead:

The overhead in the computational resources was brought about by the experience replay and the mini-batch training. By using a replay buffer, which acts as a bank of experiences (state, action, reward, next state), it means that the agent can randomly draw experiences and learn from them faster. In contrast, however, the use of large buckets tamed and supplemented by drawing allocation for training, undersole the rationale. Learning how to learn takes time. As the number of episodes increases, the duration of the replay bucket also increases, and the agent has to handle a lot of information. To change the weights in the neural network backpropagation requires repeating the batches a few times. This triad of operations—addition of replay buffer, drawing of mini-batch samples, and reinforcement of neural networks—can become taxing on memory resources rated in floating point operations, however raised training time ratings per episode or increasing batch size can influence these limitations dramatically. Management of these bonuses must be emphasized, lest the training becomes inefficient and monstrous.

Observations

Monitoring

1. Defensive Theories (Protecting from Losing Moves by Opponent):

One of the aspects which stood out o training was the capability of the agent to make defensive theories. To begin with, the agent played random games and would most of the time neither see nor prevent the ultimates of the others. With the increase in the number of training sessions, the agent understood the importance of avoiding Player 1's winning chances. However, after ~ 50 episodes, the agent figured out some strategies on the board that would mostly favor Player 1's win and started incorporating its moves in a defensive way. This improvement underscores the fact that the agent has the capability to not only plan winning moves but also defend against other player's winning moves. It is important for one to learn how to adopt such defensive strategies in order to improve agent's performance in a game, especially a competitive one; since the agent will most likely not lose because he or she will always try to stop the opponent from winning in the first place.

2. Positional Preference in the Later Stages (Central and Corner Positions):

Another major observation was the fact that the agent also began to show a preference for certain positions on the board i.e. the center and corners, as training advanced. This behavior came into play as a consequence of the agent acquiring skilled knowledge on how to play Tic Tac Toe effectively where the center and corner positions were regarded as the best positions in the game. The agent understood that these positions allowed one to create many winning lines (horizontal, vertical, diagonal) which were important to dominate the game. At the very beginning of the training period, the agent placed strat egies without any bias; as it began to understand the significance of positioning, it progressed towards moves that would favor its control of the board. The modification of the investor's strategy was driven by the agent's appreciation of the dual importance of offense (creating winning lines) and defense (preventing the opponent from winning).

2. Overfitting Caused by Ensured Protocol Training The agent in an incomplete manner of training plays with random opponent merely whose style gives a patterned way of possible responses movement arm would result to overfitting which characteristically the dominated behavior of agents who trained to confront one foe strategy. Overfitting takes place when the opponent is managed in excess, in such a case all learnt mean cannot be effectively used against other opposisions within other strategies. In case the opponent is animated such that all its movements are random, the agent within that environment can recall subject moves such that it only learns how to defeat those specific agents and fails against more complex opposition or even different levels of variability with its own opposing players . This prevents few strategies becoming dominant when agents are trained over many episodes without opponent strategy changes. To curb overfitting, it will be wise to consider a broader spectrum of strategies or to other more advanced tactical counter actions that include playing different styles of AI opponents. Such would help the agent in formulating

and polishing strategies that are more open and applicable in various situations of the game. Also, basing training on many strategies and including their evaluation frequently would control the tendency for the play of the agent to bias towards certain strategies if any.

---

5. CONCLUSION

The SQN-based agent convincingly mastered the game of Tic-Tac-Toe establishing that reinforcement learning is practical for games that have smaller scale environments. Agent was able to explore the environment and achieved the impressive win rate at evaluation thanks to the application of experience replay and epsilon-greedy policies.

Key Takeaways

- Bellman equation integration directly into the network enables effective Q-function approximation.
- Experience replay enhances training stability by diversifying learning examples.
- Epsilon decay balances the exploration-exploitation tradeoff effectively.

---

6. Suggested Improvements and Future Work

Though these results are promising, the performance may be further boosted by making improvements in multiple areas:

Diversity of the Opponent: Training against a variety of opponents, which includes strategic AI agents, should make the agent substantially more robust.

Hyperparameter Tuning: Experiment with the learning rate, the batch size, and the number of hidden neurons to better performance.

Reward Shaping: Provide intermediate rewards for strategic moves, such as creating two in a row, to encourage the learning process.

Training Episodes: Expand training episodes for greater complexity in the strategy learning process.

Generalization: Testing on modified versions of Tic-Tac-Toe or some other grid-based game can assess how well the agent generalizes.

Complexity of Models: Deeper networks or convolutional layers can be incorporated for more complex environments.

---

7. References

- Artificial Intelligence A modern Approach 4th edition
- TensorFlow Documentation.