Assignment 5a - Reinforcement Learning Analysis using Tic-Tac-Toe

Tanushree Kumar | tanushree.kumar@outlook.com

DATA 640 - Fall 2024

Professor Steve Knode

University of Maryland Global Campus

Due: October 22, 2024

Tic-Tac-Toe Q-Learning Code Overview

Q-learning is a type of reinforcement learning where an agent learns optimal actions through trial and error by maximizing cumulative rewards. In the context of Tic-Tac-Toe, Q-learning enables the algorithm to discover the best moves by repeatedly playing games, either against itself or a human opponent. The algorithm assigns a Q-value to each state-action pair, which reflects the expected reward of making a particular move from a specific board state. Through iterative updates to the Q-table, the algorithm learns the long-term value of actions, developing a strategy that increases its likelihood of winning or drawing.

For Tic-Tac-Toe, the primary purpose of Q-learning is to help the algorithm predict the best move by evaluating each action's reward potential based on past games. The exploration-exploitation tradeoff, which is using random moves vs. following known high-value actions, ensures the algorithm gathers enough information to refine its Q-table effectively. Over many episodes, the Q-learning process ultimately produces an algorithm capable of playing optimally by avoiding actions that would lead to losses, preferring those that lead to victories, draws, or favorable positions.

The Q-learning code used trains an algorithm to play Tic-Tac-Toe by updating a Q-table that records the expected reward of taking specific actions from each game state. Starting with a board initialized as a zero-filled 3x3 grid, the Q-table is also initialized to zero and then gradually populated with values as the agent learns. Each game state is converted into a unique integer state representation using 'board_to_state' to simplify indexing in the Q-table, which has dimensions corresponding to all possible board states and moves. The algorithm chooses its moves by balancing exploration, which is done by trying different moves to learn their value, and

exploitation, which is done by choosing known high-value moves, adjusting its strategy based on an epsilon-greedy approach. The epsilon starts at a maximum, gradually decaying over time to encourage more exploitation as the algorithm becomes more confident.

The ‘get_epsilon’ function manages the decay, which is visually displayed in a plot to show how exploration shifts towards exploitation over episodes. When the algorithm needs to make a move, ‘get_available_moves’ provides legal move options on the current board. The ‘is_terminal’ function then checks if the game has reached a terminal state: win, loss, or draw. If the algorithm wins or draws, it back-propagates the reward through the Q-table entries associated with the episode's actions, using the Bellman equation to update values. Each update improves the algorithm’s decisions, as it learns which moves maximize future rewards. The code then periodically records and plots the win and draw probabilities over time, as well as the sum of Q-values to illustrate the algorithm’s improvement.

The code also includes a user interaction mode where the trained algorithm competes against a human, and each turn is displayed, allowing the human to enter moves and see the algorithm’s response. The training results, saved in files, allow for further analysis, especially when the game is extended from a 3x3 to a 4x4 grid. This flexibility allows for scaling to more complex versions of Tic-Tac-Toe, demonstrating the adaptability of Q-learning to learn from larger state-action spaces.

The two plots produced in the code serve different purposes related to the agent’s training progress and learning stability. The first graph is the Win-Loss-Draw Ratio plot which shows the probabilities of draws and wins across episodes, highlighting the agent's improvement in gameplay. It also tracks the cumulative Q-values which indicate the stability and progression of learned values. As the Q-table stabilizes, this plot shows whether the agent’s strategy is

becoming consistent, signifying readiness for exploitation over exploration. The second graph is the Q-Value plot which visualizes the decay of epsilon over time, representing the transition from exploration to exploitation. The epsilon starts high, allowing the agent to try random moves, but gradually decreases, encouraging the agent to use its learned knowledge as it grows more confident.

Ryan Rudes originally wrote the code for his article on "Learning Tic-Tac-Toe via Self-Play Tabular Q-learning" (Rudes, 2020b). His code can be found in the Python file on Google Colab (Rudes, 2020a). His code was used for this analysis, with modifications made for the different minimum and maximum epsilon values and other necessary changes when extending the board from 3x3 to 4x4. Ideally, the other parameters would be untouched to assess the model purely based on the different epsilon values. However, a value of 1,000,000 is computationally intensive; thus the value was reduced to 10,000 so it is on a more manageable scale. There is a drawback to reducing the number of episodes in Q-learning as it limits the agent's learning opportunities, leading to a less comprehensive exploration of possible states and actions. Fewer episodes often result in a Q-table that is under-optimized, as the agent hasn't fully learned the best strategies for various states. This typically affects the stability of the learned policy, making the agent more prone to suboptimal or inconsistent moves, especially in complex games. With a reduced number of episodes, the agent may not generalize well to

A total of 6 models were made with the first three models for the 3x3 grid and the last three models for the 4x4 grid. The table below shows the differences and results of each model:

Model No.	Board Size	Episodes	Min Epsilon	Max Epsilon	Episode	Epsilon	Win Prob.	Draw Prob.
1	3x3	1,000,000	0.01	1.0	999001	0.010	96.900	3.100
2	3x3	100,000	0.01	0.5	99001	0.101	25.900	74.100
3	3x3	100,000	0.01	0.1	99001	0.100	89.200	10.800

4	4x4	1000	0.01	1.0	991	0.0100	12.0000	88.0000
5	4x4	1000	0.01	0.5	221	0.0100	2.0000	98.0000
6	4x4	1000	0.01	0.1	221	0.0100	11.0000	89.0000

Table 1. Table of values and results for all 6 models.

3x3 Tic-Tac-Toe Result

Higher initial epsilon values lead to more exploration initially, often showing lower win probabilities due to more random actions. A smaller final epsilon (as seen in Model 1 with $\varepsilon = 0.010$) usually implies more exploitation at later episodes, showing more stability in win rates and higher win probabilities.

For example, Model 1, with a significantly lower epsilon, displays a much higher win probability of 96.9% compared to Model 2, which has a relatively high epsilon and shows a more exploratory approach with a much lower win probability of 25.9% and high draw probability of 74.1%.

Model 1, with the lowest epsilon value, shows a high win probability of 96.9% and a low draw probability of 3.1%, indicating that the model has likely achieved an effective strategy with minimal exploration and more exploitation.

Model 2, with a higher epsilon value, exhibits a low win probability of 25.9% and a high draw probability of 74.1%, which suggests that the model is still exploring significantly, leading to more draws rather than consistent wins.

Model 3, with a similar epsilon to Model 2 but a slightly lower value, has a higher win probability of 89.2% and a lower draw probability of 10.8%, illustrating that even a slight reduction in epsilon can positively impact model performance in this setting.

Lower minimum epsilon values favor exploitation and stable Q-value convergence, as seen in Model 1, which achieves a high win rate. Conversely, higher epsilon values encourage

exploration, potentially reducing immediate win rates but may lead to a more adaptable strategy in broader applications where continuous learning and adaptability are desired.

4x4 Tic-Tac-Toe Result

Models 5 and 6 have incomplete results due to computational difficulties. In the 4x4 model cases, the outcomes differ significantly. Model 4 shows a very low win probability of 12.0% alongside a high draw probability of 88.0%, indicating a lack of effective strategy or inadequate training. This situation mirrors Model 2 from the 3x3 cases. Model 5 reveals an even lower win probability of 2.0% and an extremely high draw probability of 98.0%, suggesting almost no capacity to win, likely due to insufficient exploration or learning given the limited number of episodes (221). Model 6, while slightly better, still shows a low win probability of 11.0% and a high draw probability of 89.0%.

In the 3x3 model cases, Model 1 shows a high win probability of 96.9% and a low draw probability of 3.1%. This indicates strong performance, likely due to effective training and strategy development. Model 2, on the other hand, exhibits a lower win probability of 25.9% with a much higher draw probability of 74.1%. This suggests that the agent is either exploring too much or not learning effectively from its experiences, leading to many draws instead of wins. Model 3 presents results similar to Model 1, with a win probability of 89.2% and a low draw probability of 10.8%, again indicating effective learning and strategy development.

Epsilon plays a crucial role in the performance of these models, as it governs the trade-off between exploration and exploitation in the epsilon-greedy strategy used in Q-learning. Lower epsilon values typically encourage more exploitation, while higher values promote exploration. In the 3x3 models, those with lower epsilon (e.g., Model 1) tend to perform better, reflecting a focused approach to exploiting learned strategies. Conversely, the 4x4 models maintain low epsilon across the board, which may not be sufficient given the increased complexity of the state

space. This insufficient exploration likely contributes to the lower win probabilities and higher draw rates observed in these models.

The complexity of the game board significantly impacts the learning outcomes in Q-learning, with the epsilon parameter playing a pivotal role in balancing exploration and exploitation. Adjusting epsilon to facilitate more exploration could enhance the performance of the 4x4 models, allowing the agent to uncover more effective strategies in a more intricate environment.

References

Baeldung. (2020, December 18). Epsilon-Greedy Q-learning | Baeldung on Computer Science. Www.baeldung.com. https://www.baeldung.com/cs/epsilon-greedy-q-learning

Rudes, R. (2020a). *Tic-Tac-Toe Tabular Q-Learning.ipynb*. Google Colab. https://colab.research.google.com/drive/1w3RYXZ_tg80qNDQZf1I2KyZcigwz8Not?usp=sharing&source=post_page-----b8b845e18fe----------------------------

Rudes, R. (2020b, November 21). *An Introductory Reinforcement Learning Project: Learning Tic-Tac-Toe via Self-Play Tabular*…. Medium; Towards Data Science. https://towardsdatascience.com/an-introductory-reinforcement-learning-project-learning-tic-tac-toe-via-self-play-tabular-b8b845e18fe

Appendix A

All figures and visualizations mentioned in the report can be seen below. The python script used to generate the models and the visualizations is attached as a separate file.

Figure 1

Figure of the Win-Loss-Draw Ratio and Q-Value plot for Model 1.

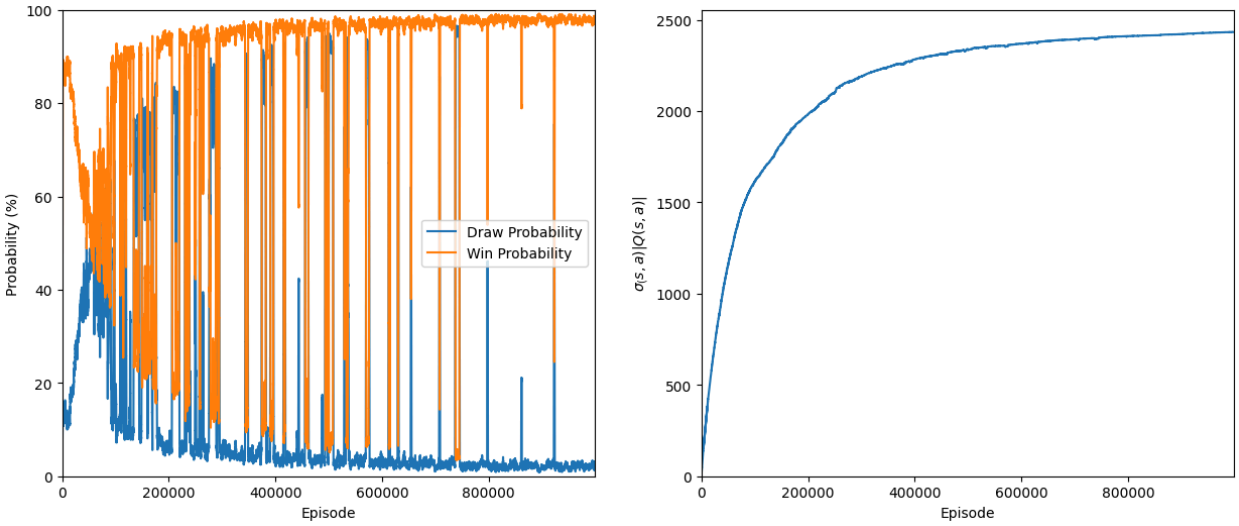

Figure 2

Figure of the Win-Loss-Draw Ratio and Q-Value plot for Model 2.

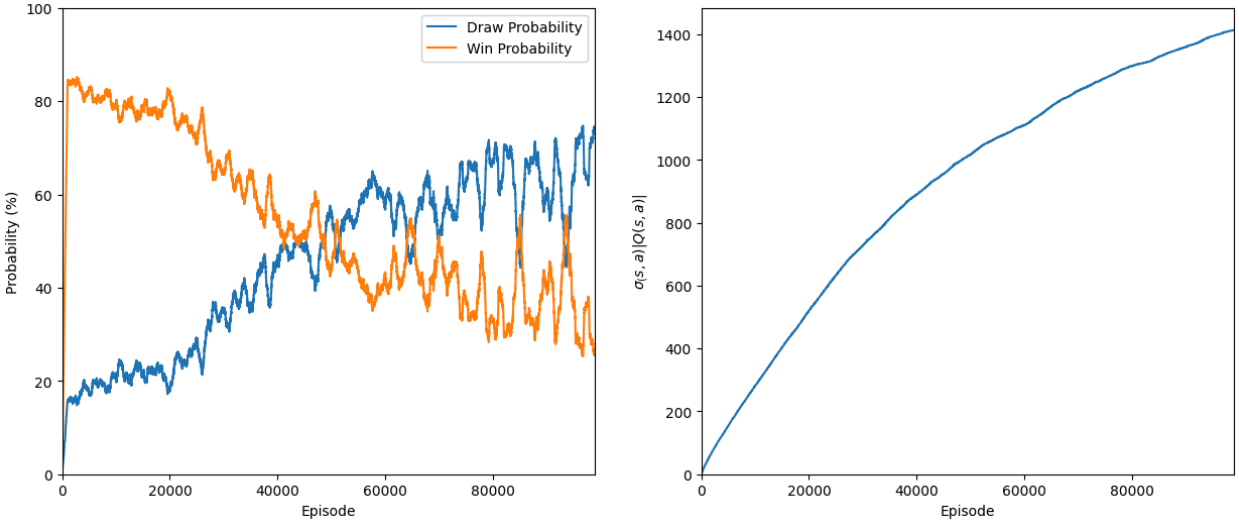

Figure 3

Figure of the Win-Loss-Draw Ratio and Q-Value plot for Model 3.

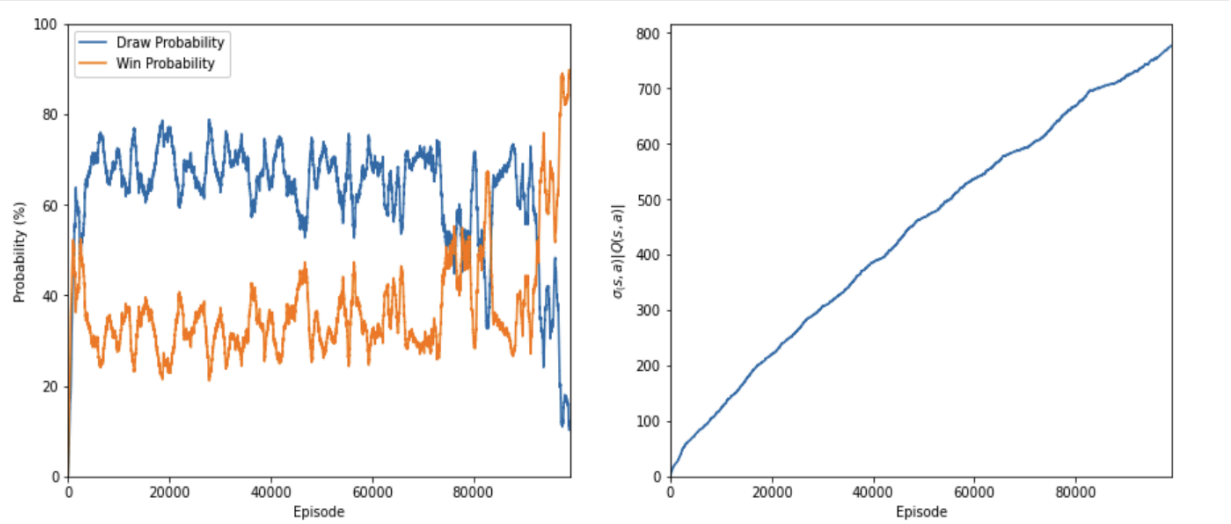

Figure 4

Figure of the Win-Loss-Draw Ratio and Q-Value plot for Model 4.

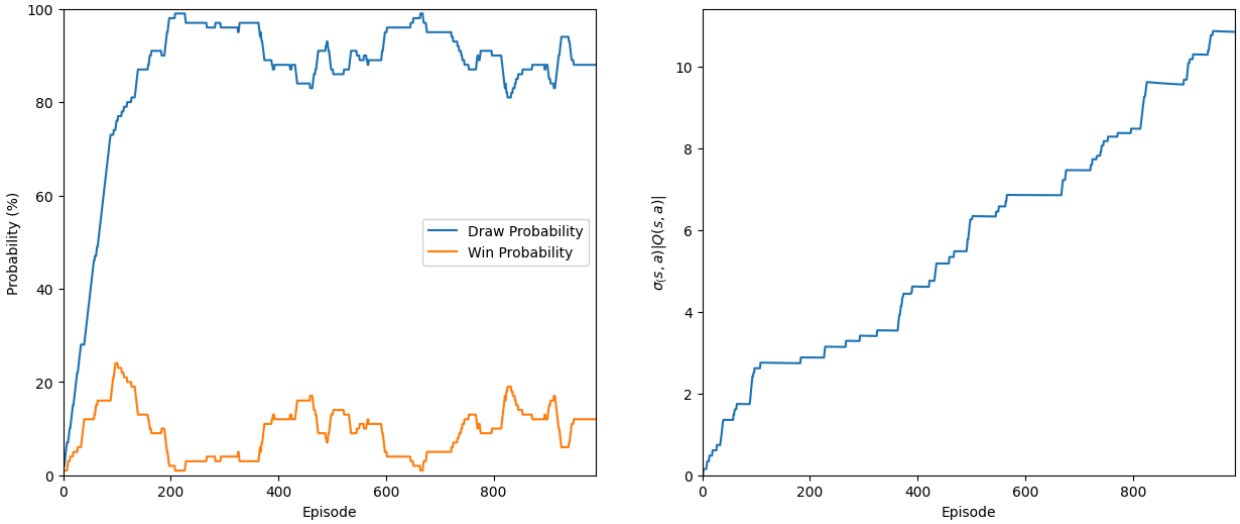

Figure 5

Figure of the premature Win-Loss-Draw Ratio and Q-Value plot for Model 5.

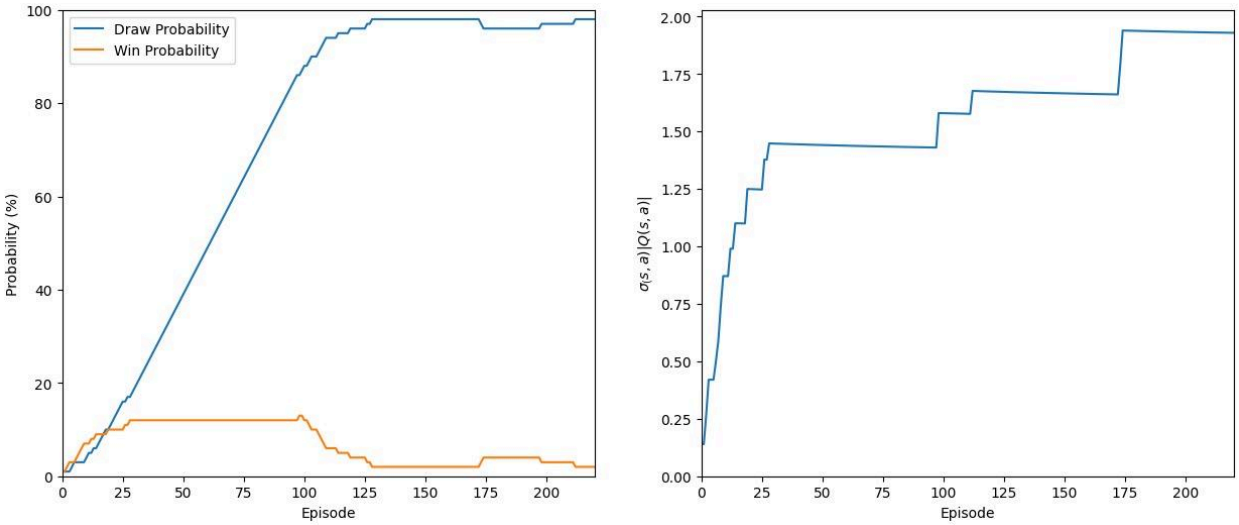

Figure 6

Figure of the premature Win-Loss-Draw Ratio and Q-Value plot for Model 6.