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

Theodore Fitch
Department of Data Analytics, University of Maryland Global Campus

DATA 640: Predictive Modeling

Dr. Steven Knode

March 19th, Spring 2024

Tic-Tac-Toe Q-Learning Code Overview:

In this brief report, we'll demonstrate Tabular Q-Learning (a form of Reinforcement Learning) where a model-free algorithm will learn strategies playing Tic-Tac-Toe against itself first in a classic 3x3 grid and then in a 4x4 grid. Several common Reinforcement Learning (RL) terms are defined in Table 1 and will be used throughout the report. The original code can be seen at Rudes (2020) or the code modified for this analysis can be found at Fitch (2024). An overview of all generated iterations of the algorithm are found in Table 2. All coding was performed using Python in Google Colab. Q-learning is a type of Reinforcement Learning in which an agent learns to maximize a Q-value which represents future rewards for actions taken in its environment (Q(s,a) representing the Q-value of a particular state and action)(Sutton and Barto, 2017). A high-level summary of the code will briefly be given. First, the needed libraries are imported and Q-table is defined. Then the hyperparameters are set including number of episodes, learning rate, number of random episodes, and minimum/maximum Epsilon (ε). This last range is key to this exploration as ε denotes how likely the algorithm is to Explore a new strategy (by choosing a random action) to potentially increase reward versus Exploiting what it knows in order to gain maximum immediate reward. Accordingly, an ε decay function generates the training strategy such that as ε decreases, less random actions are taken. This moves the agent's understanding from first exploring the environment using random actions to falling back on what it knows to win. The board state is created (3x3 grid) with 3 possible states: null, X, or O. Code is written to determine if the board is ever in a terminal state with 3 X's or O's in a row, column, or diagonal. The Q-Learning is generated where the main loop is defined (using the set episode number). Each episode, the agent interacts with the environment (the computer playing the Tic-Tac-Toe) and the Q-values are updated based upon the corresponding received rewards for each action it takes (and thus it learns strategies). The agent always aims to maximize the Q-value.

Lastly, the code generates two plots: 1. A plot to show the probability of Win vs Draw over each episode 2. A plot showing the increase in Q-value (absolute value) over episodes. As the Q-value reaches an asymptote (convergence), this demonstrates the agent has learned an optimal policy. It’s critical to mention that this code only focuses on wins and draws instead of losses. Losses are automatically accounted for as there are only 3 possible outcomes: if wins increase, and draws decrease, then the agent implicitly decreases losses as well.

3x3 Tic-Tac-Toe Results:

The experiments involved 4 iterations of the Q-learning algorithm each using various exploration strategies (Table 2). First, the baseline code was run with all the same hyperparameters as the original analysis using an ε of 0.01 min to 1 max (Rudes, 2020)(QL1: Figure 1). This produced the results where the agent explored strategies and started with a relatively low win rate in the first 100,000 episodes but slowly increasing. The agent appears to reach around a 90%-win rate near 150,000 episodes and this slowly increases to near 100% with many sudden drops in between. This shows how the agent begins Exploring the environment and new strategies since it had a large range of ε. The decay function forces the agent to rely more heavily on Exploitation as episodes increase which is why we steadily see an increase in win rate, and some level of convergence; but there are still many times where the agent attempts to Explore and the plot shows sudden jumps between the win and draw probabilities. In order to demonstrate the effects of ε, QL2 was generated with an ε range of 0.01-0.05 (Figure 2). This shows how the agent took more time to Explore up to about 200,000 episodes with high fluctuations in win rate to find convergence after this. Compared to QL1 with many significant fluctuations throughout, QL2 only contained 6 observable fluctuations in the win rate. It was a much more conservative approach as the agent took longer to Explore and relied more on Exploitation. This is similarly reflected in the Q-value plots where convergence is observed around episode 800,000 for QL1 but 600,000 for QL2. QL1 and QL2 had somewhat similar overall learning rates (~2,00 and 2,000) and this is likely because the ε decay function makes the agents lean more towards Exploitation. In contrast, QL3 was generated with an ε range of 0.05-1 to demonstrate the effects of forcing more Exploration (Figure 3). This agent never had fluctuations between high/low win rates and always maintained a high win rate clearly due to exploring the environment quickly and continuously learning strategies. The lowest it fell was slightly past 80% win at <50,000 episodes but always sustained a ~80% win probability. The Q-value plot interestingly did not show convergence occurring (it was clearly still increasing and had not reached an asymptote). This is potentially due to the agent constantly exploring the environment and continuously finding new strategies (by exploiting its opponent's mistakes). As a final iteration of the 3x3 game, QL4 was generated using an ε of 0.1-0.1 to demonstrate using a fixed ε (Figure 4). The resultant plots show ε's effects most clearly with the agent starting with only a ~10% win rate up until episode 400,000. Strong fluctuations are then observed between 400,000-650,000 episodes until it essentially reaches convergence where the win probability far outpaces the draw probability. The win rate pushes up towards nearly 100% at episode 1,000,000. The Q-plot also demonstrates that the agent was reaching convergence as the shape appears logistic and it is reaching its asymptote. The values also demonstrate how thoroughly the environment was explored where QL4 asymptote appears roughly near 800 (low ε, low Exploration), QL3 appears near 12,000 (high ε, high Exploration), and QL1/QL2 appear near 2,500/2,000 respectively. Since Q-values reflect how the agent is learning, each of these models reflect their respective ε strategies to Explore more or Exploit more. Thus, these agents have reflected exactly how we'd expect them to act by adjusting their ε ranges.

4x4 Tic-Tac-Toe Results:

The code was then adjusted to expand the game from a 3x3 typical game of Tic-Tac-Toe to a 4x4 grid. This makes the environment much more complicated: with 3 possible states of each square and 16 squares, this is 3^{16} = 43,046,721 possible game states (compared to 19,683 for a 3x3 grid). This slows down computing tremendously thus only 1,000 episodes were used (and other parameters were attuned appropriately). QL5 was generated with an ε range of 0.01-1 and showed several key differences from QL1 (from which it was based)(Figure 5). First, it appeared to converge with an extremely high rate of draws instead of wins. Over the entire 1,000-episode series, win probability never outpaced draw probability. Second, the Q-value stayed exceptionally low only reaching a max of 6 meaning the agent did not learn well. To confirm this, I watched the output ε every episode plot early on while the program ran; it showed the epsilon decayed immediately to 0.01. This indicated it did not take the time to properly train, exploring the environment to learn more strategies to win, and instead relied too heavily on Exploitation. In order to compensate for this observation, QL6 was made which had an ε range of 0.5-1 such that it manually forced the agent to Explore more (Figure 6). QL6 had a Q-value >50 and was not reaching its asymptote. It appeared to still be growing linearly meaning more train time would be needed to find its convergence point. Within the first 100 episodes, the win and draw probabilities traded places many times as they steadily increased towards ~50%. Then, from episode 100-1,000 the rates stay within 40-60% with win probabilities staying higher than draw for almost all of the duration. It would be interesting to give this agent more time to allow it to find an asymptote to Q-values. It is unexpected that QL6 was so different from QL3 but this is likely due to the nature of 4x4 Tic-Tac-Toe. As mentioned, it is significantly more complicated and likely needs more training time for the agent to be successful. This is counterbalanced by the training being computationally expensive and needing to find the equilibrium between training towards accuracy and training time. Further training could be performed on QL6 in order to determine when it reaches convergence.

References:

Fitch, T. (2024). *RL-Tic-Tac-Toe*. GitHub. https://github.com/Capadetated/RL-Tic-Tac-Toe

FreeCodeCamp. (2018). *An introduction to Q-learning: Reinforcement learning*. FreeCodeCamp. https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/

Rudes, R. (2020). An Introductory Reinforcement Learning Project: Learning Tic-Tac-Toe via Self-Play Tabular Q-learning. TowardsDataScience. https://towardsdatascience.com/an-introductory-reinforcement-learning-project-learning-tic-tac-toe-via-self-play-tabular-b8b845e18fe

Sutton, R. & Barto, A. (2017). Reinforcement Learning: An Introduction. The MIT Press. http://incompleteideas.net/book/bookdraft2017nov5.pdf

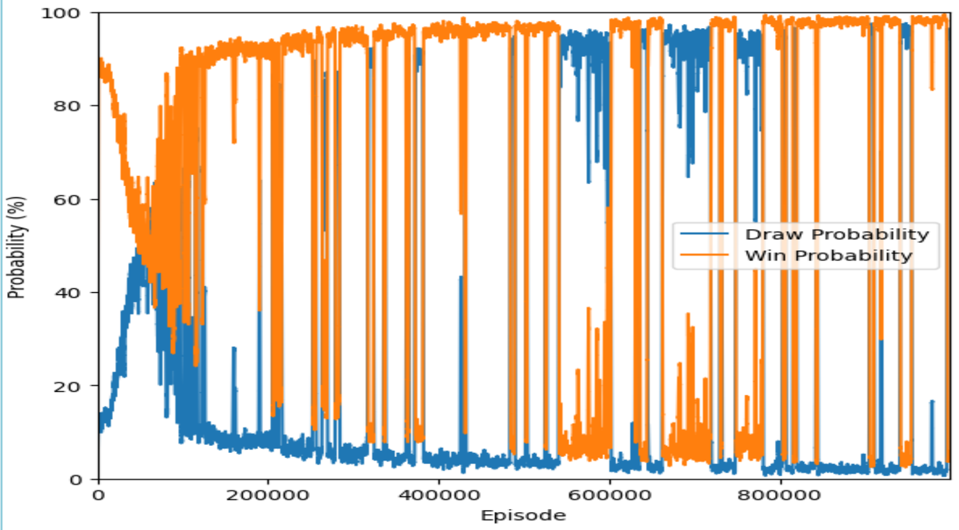
Appendix:

Term	Definition
Agent	The learner or decision-maker interacting with the environment.
Environment	The external system with which the agent interacts.
Actions	The set of possible decisions or moves that the agent can take.
State	The current situation or configuration of the environment.
Rewards	Numeric feedback from the environment indicating the desirability of the agent's action.
Policy	The strategy or mapping from states to actions that the agent follows to maximize cumulative reward.
Value Function	An estimate of the expected cumulative reward that an agent can obtain from a given state or state-action pair.
Q-Learning	A model-free RL algorithm that learns a Q-value function representing the expected future rewards of taking a particular action in a given state.
Q-Value	A value of expected future rewards for the agent of taking a particular action in a given state.
Exploration vs. Exploitation	The trade-off between trying new actions to discover better strategies (Exploration) and exploiting known actions for immediate reward (Exploitation).
Epsilon (ε)	The probability of randomly selection actions to explore new possibilities ranging from 0 to 1 inclusive.

Table 1. Definitions for common reinforcement learning terms needed to understand this analysis (Sutton and Barto, 2017).

Index	Model Name	Min ε	Max ε	Game Size
1	QL1	0.01	1	3x3
2	QL2	0.01	0.5	3x3
3	QL3	0.5	1	3x3
4	QL4	0.01	0.01	3x3
5	QL5	0.01	1	4x4
6	QL6	0.5	1	4x4

Table 2. Overview of the 6 Tabular Q-learning algorithms generated to play Tic-Tac-Toe.



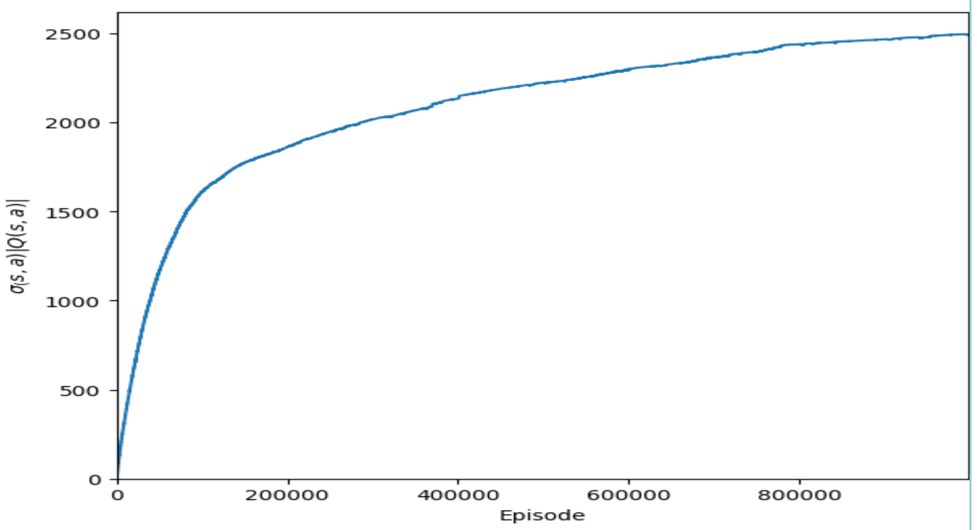
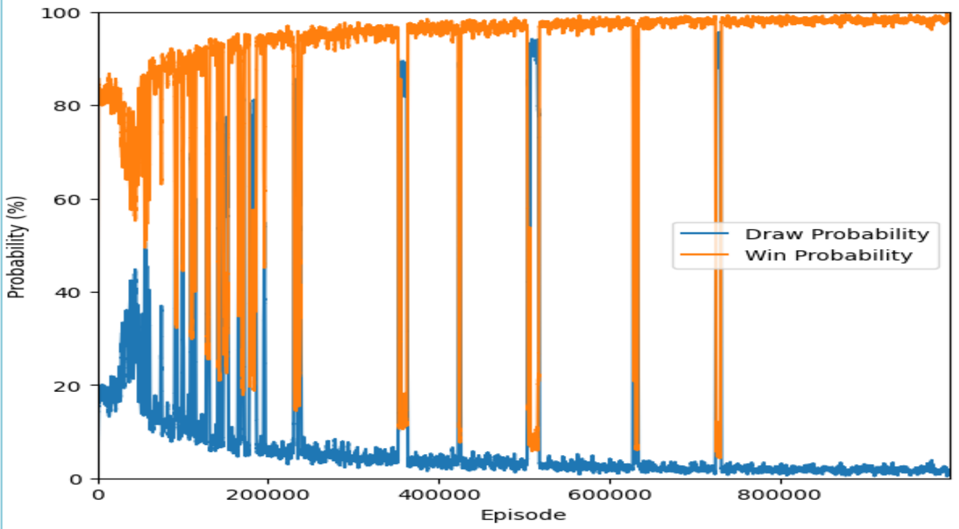


Figure 1. Win and draw probability plot and Q-value plot for QL1 using baseline parameters and an ε of 0.01 min to 1 max.



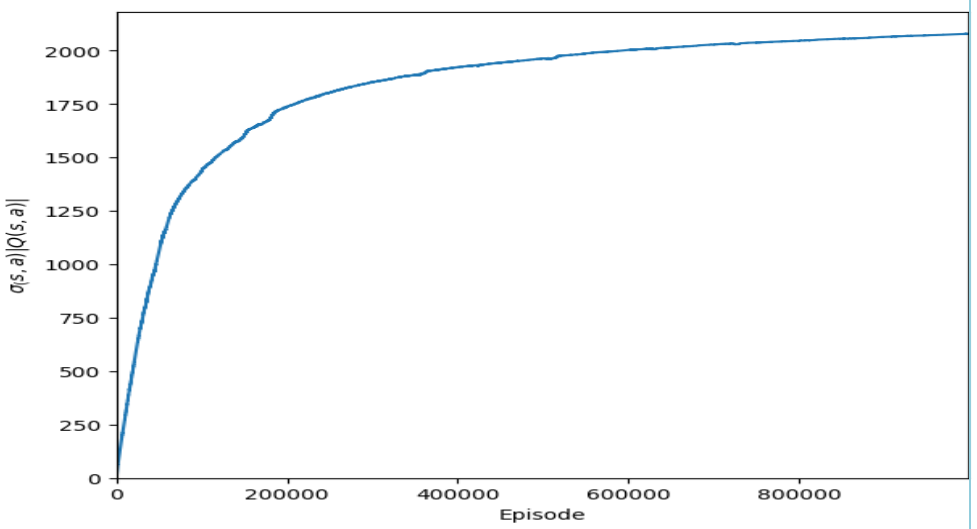


Figure 2. Win and draw probability plot and Q-value plot for QL2 using an updated ε of 0.01 min to 0.5 max.

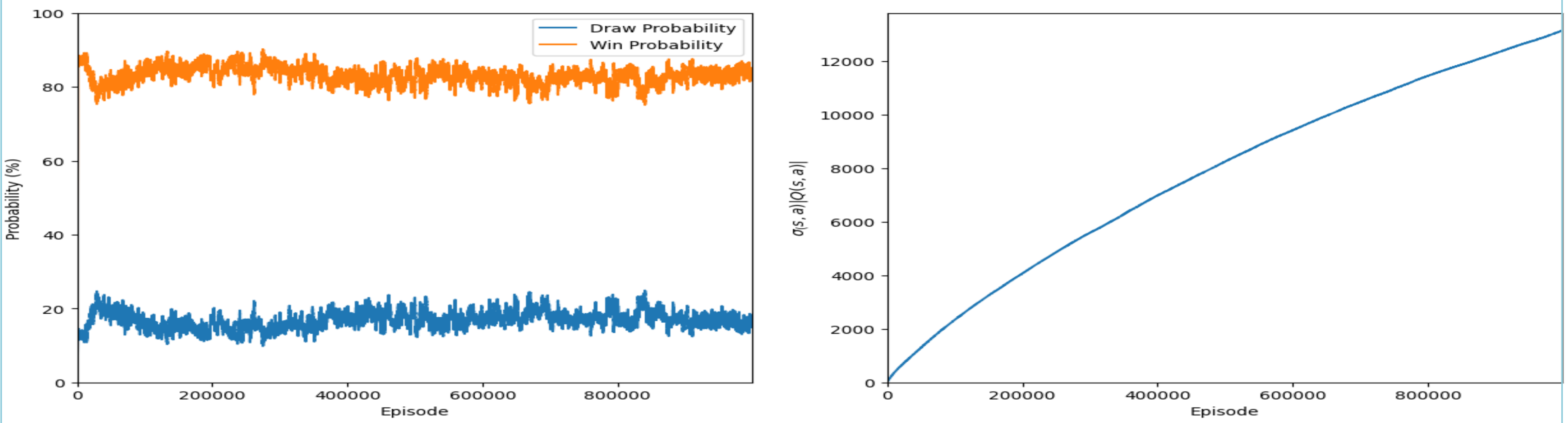


Figure 3. Win and draw probability plot and Q-value plot for QL3 using updated ε of 0.5 min to 1 max.

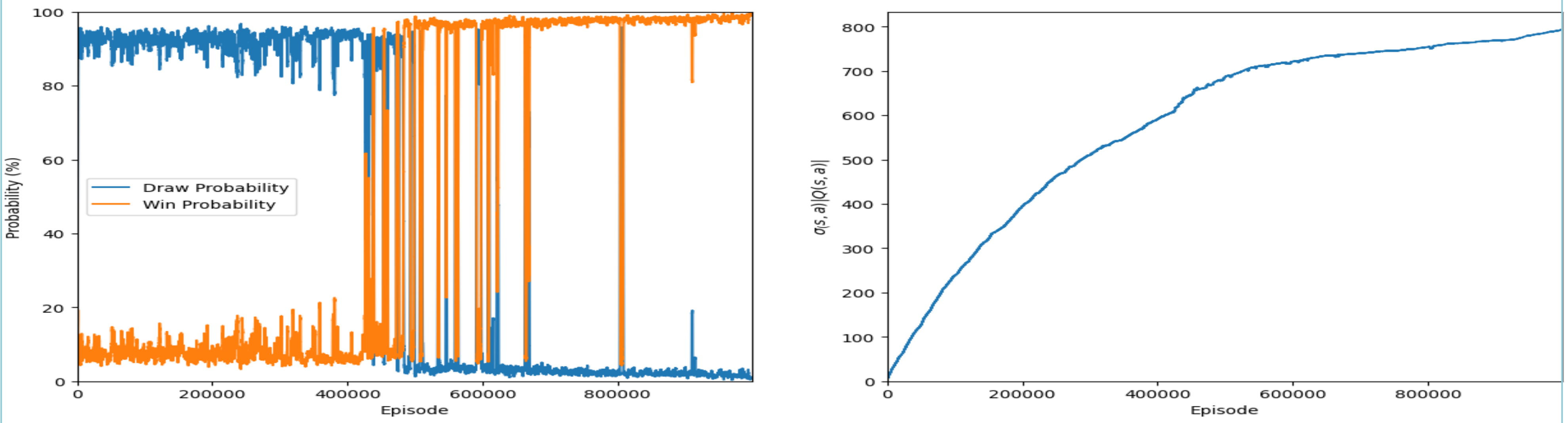


Figure 4. Win and draw probability plot and Q-value plot for QL4 using updated ε of 0.1 min to 0.1 max.

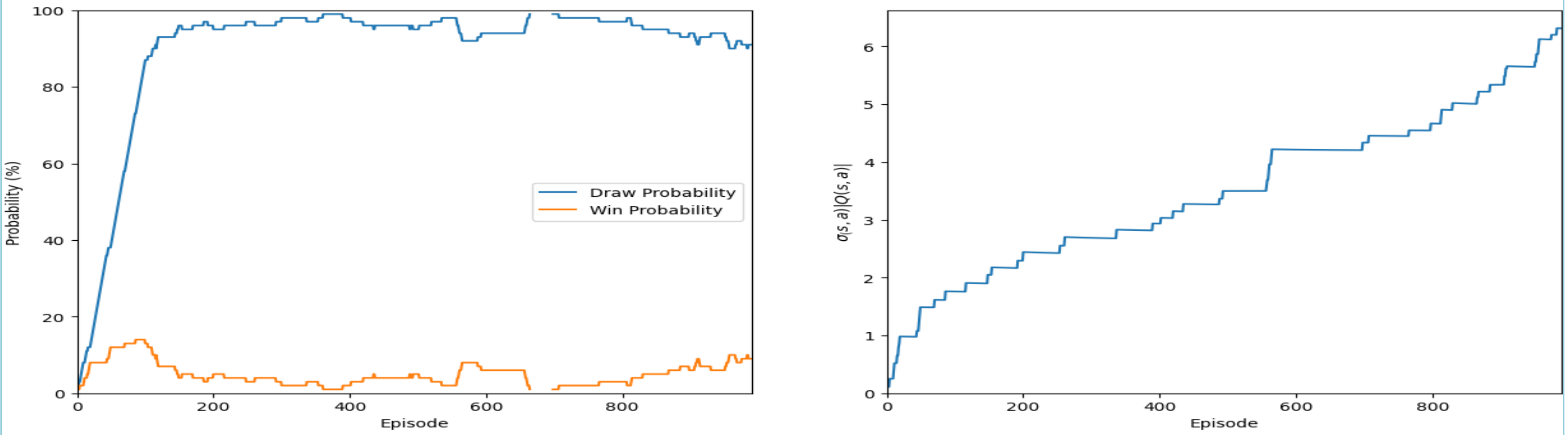


Figure 5. Win and draw probability plot and Q-value plot for QL5 using ε of 0.01 min to 1 max.

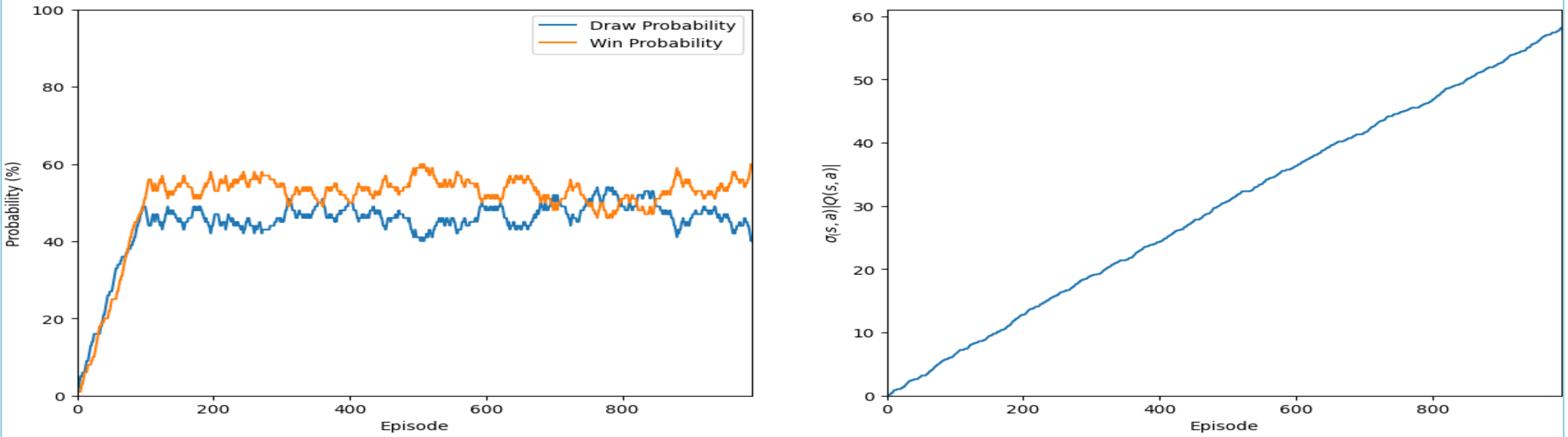


Figure 6. Win and draw probability plot and Q-value plot for QL6 using ε of 0.5 min to 1 max.