Q-learning works in the context of our Tic Tac Toe agent:

1. The Learning Process: Q-learning is based on learning a value function (Q-function) that estimates the expected future reward for taking an action in a given state. In our implementation, this is represented by the Q-table, which maps state-action pairs to values.

The core of the learning happens in the Q-learning update formula:

Copy

Q(s,a) ← Q(s,a) + α[r + γ max Q(s',a') - Q(s,a)]

Let's break this down:

* Q(s,a) is the current estimate of the value for taking action 'a' in state 's'
* α (learning rate) determines how much we update our estimate (0.1 in our code)
* r is the immediate reward we get
* γ (discount factor) determines how much we value future rewards (0.95 in our code)
* max Q(s',a') is our estimate of the best possible future value

1. The Exploration-Exploitation Trade-off: Our agent uses an epsilon-greedy strategy to balance exploration (trying new moves) and exploitation (using known good moves):

* With probability ε (epsilon), choose a random action (exploration)
* With probability 1-ε, choose the action with highest Q-value (exploitation)
* We start with ε = 0.1 and decay it over time, so the agent explores more early in training

1. Reward Structure: The rewards are carefully designed to guide the agent toward winning:

* +1 for winning: Primary goal
* -1 for losing: Learn to avoid losing positions
* +0.5 for drawing: Better than losing but worse than winning
* -10 for invalid moves: Quickly learn the rules
* 0 for ongoing moves: Focus on the end result

1. Performance Tracking: I've added performance monitoring to help visualize how the agent improves over time:

* Tracks wins, losses, and draws
* Calculates moving average of rewards
* Plots learning progress over time

The learning curve typically shows:

* Initial phase: Lots of random play, low average reward
* Rapid improvement: Agent discovers winning strategies
* Plateau: Agent reaches optimal or near-optimal play

Would you like me to explain any aspect in more detail? For instance, we could:

1. Analyse specific game situations and how the agent learns from them
2. Explore how different hyperparameters (learning rate, discount factor) affect learning
3. Look at how the state representation affects the agent's ability to learn
4. Discuss potential improvements to make the agent stronger