AI Model Evaluation: Tic-Tac-Toe

by Turner DeMott

In this report, I reflect on the behavior of my program after confirming its correctness. I evaluated how well it learned by testing different combinations of betaReward and betaPunish. I tried five different parameter sets and documented the outcomes extensively. Below is a ranked list of how effectively the machine appeared to learn with each beta combination:

1. Different beta values: R = 0.5, P = 0.25
2. Early observations: R = 0.5, P = 0.5
3. Reward / Inaction: R = 0.5, P = 0
4. Lower beta values: R = 0.25, P = 0.25
5. Higher beta values: R = 0.75, P = 0.75

Early Observations

betaReward = 0.5, betaPunish = 0.5

From the start, I played seriously rather than going easy on the AI. I easily won the first several games. Then, surprisingly, the machine began making a series of lucky but smart moves-blocking my attempts to win—and earned its first rewards.

After that, its gameplay improved rapidly. It consistently started in the center, and I could no longer beat it. The best I could do was force a draw by playing a corner first and sticking to classic strategy. If I deviated slightly—for example, starting on a side square and playing optimally from there—the AI often won.

Examining the STM, I saw that it had effectively learned to block some winning threats, but many board situations remained unexplored. I eventually figured out how to beat it—by making suboptimal moves that led to board states it hadn't encountered. For instance, it missed obvious chances to win because it hadn't learned to recognize them. In the end, the trick to beating the AI was playing counterintuitively.

Lower Beta Values

betaReward = 0.25, betaPunish = 0.25

With both betas set low, training was noticeably slower and more frustrating. It took many, many games just to get a tie, then plenty more for another tie. Eventually, it tied me a few more times when it started in the center—but it still tried other starting positions occasionally. Oddly, it once managed to beat me after starting in a corner, which shifted the STM to favor corner-first openings (possibly because I had punished many of its center-first games due to poor mid-game decisions).

The learning here was painstakingly slow, and good moves were often erased by a single poor outcome. Because the AI treats an entire game as either good or bad, even if it made several correct moves, losing still meant full punishment. I considered letting it win when it made strong moves to guide its learning—but that would turn it into supervised learning, which wasn't the intent. So, I kept playing to win and observed whether it could adapt on its own.

Higher Beta Values

betaReward = 0.75, betaPunish = 0.75

With high beta values, the AI behaved erratically at first—trying many different openings. After one game where it forced a tie with a side-square opening, it latched onto that strategy. I assumed it would stick with it, but a single loss immediately caused a major shift in STM, pushing it to favor center starts again.

A pattern emerged: the AI avoided any strategy that failed even once. It cycled randomly through opening moves, never committing long enough to improve. The steep punishment caused it to abandon strategies too quickly. It didn’t learn—it just scrambled for a move that wouldn’t lead to punishment. I

stopped this trial when it became obvious that the AI was in a constant state of confusion, unable to commit to or learn from any strategy.

Reward / Inaction

betaReward = 0.5, betaPunish = 0

This time, I removed punishment altogether. Initially, the AI learned nothing from its losses and made completely random moves. But soon, by sheer chance, it managed to beat me with a side-square start.

That accidental win caused it to focus on the side-opening strategy. Since there was no punishment, it had no reason to unlearn anything. Even if I defeated it repeatedly, its commitment to the successful path remained strong. The only way to shift its behavior would be to randomly stumble upon a better winning strategy—unlikely given the imbalance in STM weights.

This version of the AI became terrifyingly persistent. It mastered the side-start strategy and refused to abandon it, even in the face of repeated failures. Although I could still beat it by creating board states it hadn't seen, I knew it was just a matter of time before it covered all the gaps.

The AI clearly learned well from positive outcomes. But without any ability to reflect on or adapt to failure, its learning was limited and rigid.

Different Beta Values

betaReward = 0.5, betaPunish = 0.25

This final combination produced the most balanced and effective AI. After about 10 games, it started to improve steadily, gravitating toward a center-first strategy and refining it over time.

I experimented with tricking it by introducing strange board states. This sometimes worked, but the AI still managed to generalize its strategy and

respond well to different variations. Even when I forced it to lose several games in a row, it didn't abandon good strategies—it adjusted and improved.

Eventually, this AI variant became difficult to beat. If I played standard strategy, it usually won. My only chance was to play unpredictably and hope to catch it off guard. It was both frustrating and impressive.

This version aligns well with what we discussed in class: that moderate punishment and stronger rewards lead to the most effective learning. It adapts but doesn't overreact.

Conclusion

Playing against this AI transformed my view of Tic Tac Toe. I had always seen it as a solved game—predictable and uninteresting once basic strategy is understood. But playing against an unsupervised learning system shifted the paradigm.

I learned to exploit the AI's blind spots. Beating it required identifying the limits of its experience and maneuvering into unfamiliar board states. As the AI improved, so did my strategies. It became a mental duel: I had to outthink a machine that was constantly evolving.