Team $GME (Grady McPeak, Marshall Thompson, Matthew Berning)
3/26/2021

Project Report - Project 3

For our third project, we created an agent that is designed to play Tic-Tac-Toe on an *N*x*N* board with a target of length *M*, where $M \leq N$. The use of the term "target" here is the same as what was used in the Project 3 specification.

The core of our agent is a value-assignment system which we apply to each square on the board when determining where our agent should make the next move in the game, focusing on estimating how crucial a given spot would be for a win for either our agent or our opponent's agent. More specifically, for any given spot *S,* we measure the value of *S* by calculating how many other spots are nearby in each direction, with a maximum distance of the value of target. This means that for any given spot *S*, we are calculating 8 values:

1. $S_{\text{v-a}}$, the number of our agent's marked spaces either above or below *S*
2. $S_{\text{h-a}}$, the number of our agent's marked spaces to the left or the right of *S*
3. $S_{\text{d-p-a}}$, the number of our agent's marked spaces diagonally intersecting *S* in a positive slope
4. $S_{\text{d-n-a}}$, the number of our agent's marked spaces diagonally intersecting *S* in a negative slope
5. $S_{\text{v-o}}$, the number of our opponent's marked spaces either above or below *S*
6. $S_{\text{h-o}}$, the number of our opponent's marked spaces to the left or the right of *S*
7. $S_{\text{d-p-o}}$, the number of our opponent's marked spaces diagonally intersecting *S* in a positive slope
8. $S_{\text{d-n-a}}$, the number of our opponent's marked spaces diagonally intersecting *S* in a negative slope

Our agent will then assign this space's value as the highest one of these 8 sums, and the space with the highest sum will be the one that the agent selects for its next turn. This enables our agent to identify the optimal next move by finding which space is currently the most valuable for the task of getting "*M*-in-a-row" and winning the game.

Our agent also employs two notable tiebreaker calculations to boost its effectiveness. The first of these is a rather simple adjustment; if our agent determines that it is one spot away from winning, it will mark the relevant spot with a value of infinity ("`float('inf')`" in Python) in order to ensure that, when our agent is in the position to win the game, it will take advantage of the opportunity and secure the win. The second of these is a tie-breaker between two spaces whose maximum values are the same. Specifically, if the max values of spaces S^1 and S^2 are equal, then the agent will see *how many* of the 8 calculated values are equal to the reported maximum value. So, if $S^1_{\text{v-a}}=5$ but $S^2_{\text{v-a}}$ and $S^2_{\text{d-n-a}}$ both also equal 5, then we know that while S^1 is involved in one possible win, S^2 is involved in *two* possible wins, which makes S^2 more valuable than S^1, and thus a smarter move.