Probabilistic Adversarial Game Agent : Proximity

CS5100 Project Proposal

Matthias Denu and Donald Hamnett

March 5, 2018

Problem Description

Over the last few decades, the Artificial Intelligence (AI) field has made significant progress in matching and exceeding human performance in various games. Notable examples of this include IBM's Deep Blue [1] defeating chess world champion Kasparov in 1997, and more recently DeepMind's AlphaGo [2] defeating world champion Go player Lee Sedol in 2016. The goal of this project is to replicate such success, albeit on a more modest level, in regards to the probabilistic Go-like game Proximity [3].

Proximity

The objective of the game is to outscore the opponent by winning territories. A player's score is the sum of the values on their won territories. A player wins a territory by placing one of their tiles on an unclaimed territory. On placing a tile, adjacent friendly territories will have their value increased by one point. Adjacent enemy territories will be captured if they have a lower value than that of the newly placed tile. Captured territories change color to that of the capturing player. Players take turns placing tiles to capture territory. Players receive a tile of a random value between 1 and 20 on each turn. The game is finished when there are no more unclaimed territories[3, 4].

Strategy

Expectimax Search

As this is a probabilistic game, our first inclination and main strategy is to build an agent which utilizes Expectimax search [5]. The nature of Proximity raises an interesting problem to solve, in that there is a duality to the probability associated with the game: not only is the opponent's move uncertain, but the effectiveness of a given move is not guaranteed. This leads to a challenge regarding creating an evaluation function which will maximize the probability of a win based on both factors.

Other Strategies

We see two more strategies as candidates for developing a sophisticated Proximity agent. The first is a deep feed-forward artificial neural network (ANN) [6] and the second is a reinforcement learning strategy[7]. Both ANNs and reinforcement learning were used in DeepMind's AlphaGo, in coordination with probabilistic search of the state space. In order to implement these models in the same language as the game itself, we plan to use the *deeplearn.js* JavaScript framework [8]

Anticipated Challenges

One luxury which the aforementioned DeepBlue and AlphaGo had which we do not is the resources and data available to them. This poses a challenge in terms of training our more advanced models. Ideally, we would be able to test against skilled players with different strategies of play, however, currently our models will train against the provided rudimentary AI agent. This is an issue we will attempt to address by adding variability to the AI agents, but we are aware this may be difficult to overcome. Thus, we will keep our goal modest and simply aim to outperform the Proximity's current AI agent.

Results

We have not completely developed full evaluation metrics, but initially we plan to measure the wining percentage of Proximity's current AI agent against itself in a number of trials, and will later compare this performance to our model's performance against this agent.

This Is The Game

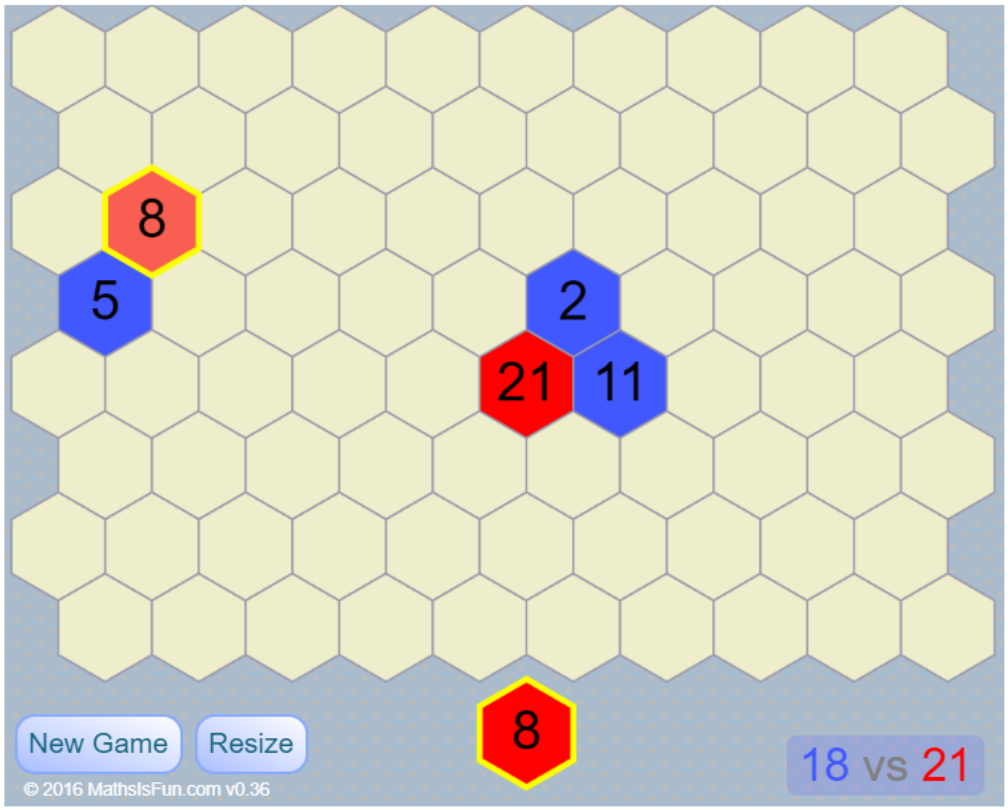

Figure 1: Proximity.

References

[1] Feb. 2001. URL: https://www.research.ibm.com/deepblue/home/html/b.html.

[2] *AlphaGo*. URL: https://deepmind.com/research/alphago/.

[3] *Proximity*. URL: http://www.mathsisfun.com/games/proximity.html.

[4] Bacable. *bacable/ProximityBasic*. Sept. 2015. URL: https://github.com/bacable/ProximityBasic.

[5] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2010.

[6] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2012.

[8] *deeplearn.js*. URL: http://deeplearnjs.org/.