CS 221 Project Proposal

Zhiyang He hzyjerry@stanford.edu
Charles Lu clu8@stanford.edu
Stephen Ou sdou@stanford.edu

October 20th, 2015

Released in 2014, the game 2048 quickly became a worldwide online sensation. In addition to hijacking classrooms around the world, it also spawned countless spin-offs, strategies, and self-professed gurus. Meanwhile, the game also offers an excellent platform to experiment and compare both traditional artificial intelligence strategies as well as cutting edge techniques.

2048 is a single player game on a 4 by 4 grid. Every turn, a random tile, with a value 2 or 4 will appear on an empty cell. Then the player can choose to slide left, right, up, or down. All the tiles will slide all the way towards that direction. If two tiles collide and have the same value, they will merge into one tile, and its new value will be the sum of the previous two. There are two ways to end the game. If the value of one tile reaches 2048, the game is considered a win. However, the player can continue to play. If all the cells on the grid are occupied with tiles, and there are no adjacent cells with the same value, the game is lost.

Our ultimate goal for this project is to build an AI program that can maximize our score at playing 2048 at a reasonable runtime. The input-output behavior is given a 2048 board and an adversary which randomly inserts a new tile after each move. On each move, our models will return one of four moves to attempt to maximize the score.

We can apply a varied set of artificial intelligence techniques to play the game. We will begin by framing the game as a Markov decision process, focusing on algorithms such as expectimax trees and analyzing their advantages. However, due to the nature of such search algorithms, they often require a base level of computation to arrive at the optimal move. Knowing this, we will apply further advanced techniques such as pruning to decrease the search space. We will then change direction and attempt to play the game using an array of machine learning and reflex methods, training these models using our results from search. By using such models, we hope to capture intuition in the game playing process and greatly diminish computation time. Yet further, we plan to investigate the use of reinforcement learning in training an artificially intelligent 2048 player and compare its performance with other techniques. Finally, we will attempt and evaluate using perhaps ludicrous state-of-the-art techniques

like deep Q-learning (in the fashion of Google DeepMinds breakout player) by feeding images, rather than raw data, of the game.

In terms of the baseline performance, we can randomly choose a move on each turn. The expected score in this case is 107.32 (based on 5 million simulations). If we were to try to emulate more humanlike random playing by randomly choosing a move different from the previous move on each turn, the expected score only increases to 121.23. Meanwhile, the minimum score to form a 2048 tile is 18432 (assuming only 4 point tiles are spawned). Another popular approach by amateur players is to alternate moving between two directions, i.e. left, up, left, up.

The performance of our algorithms can also be compared to an oracle search algorithm which is able to cheat by knowing deterministically the value and location of the next spawned tile given a board and move.

Due to the fact that 2048 is a popular game and is also easily modeled to a state-based problem, people have built AI bots to solve the game. ov3y used iterative deepening depth-first alpha-beta search to solve 2048 (https://github.com/ov3y/2048-AI). Maarten Baert used a minimax-based AI to solve 2048 (https://github.com/MaartenBaert/2048-ai-emscripten).