CS 221 Project Proposal
Building a 2048 Solver

Team: Zhiyang He, Charles Lu, Stephen Ou
SUNetIDs: `hzyjerry, clu8, sdou`

October 20th, 2015

Released in 2014, the single-player puzzle game 2048 created a stunning global hit: in addition to hijacking classrooms and quickly occupying peoples phone screens, it also spawned countless spin-offs, strategy guides, and self-professed gurus. Meanwhile, the game offers a good platform to experiment and compare both traditional and cutting edge artificial intelligence 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 any two tiles collide and have the same value, they will merge into one tile, with a new value being the sum of the two old tiles. Additionally, when two tiles collide, the score will be incremented by the value of the newly merged title. There are two ways to end the game. If the value of one tile reaches 2048, the game is considered a win (though the player can continue playing). Otherwise, if all the cells on the grid are occupied with mismatched tiles and no move is possible, the game is lost.

Our goal for this project is to build an AI program that maximizes 2048 game score 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 (left, right, up, down) to attempt to maximize the score.

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 [1]. Meanwhile, the minimum score to form a 2048 tile is 18432 (assuming only 4 point tiles are spawned) [2]. 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. By running a search algorithm in $O(4^n)$ time, it will find the optimal way to reach the highest possible score. The oracle algorithm is allowed to reach up to $2^{17} - 2$ game score by filling up all the grids with highest possible values, which is approximately 1100 times the score of baseline algorithm.

We will apply a varied set of artificial intelligence techniques to play the game. To begin with, we will frame 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 a set of different machine learning and reflex methods, which are trained using our results from search. By applying these models, we hope to capture intuition in the game playing process and save computation time as much as possible. Furthermore, 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 Atari player) [3] by feeding images, rather than raw data, to the game.

To help build our AI, we will learn from existing projects done by other people, most of which are available online. An anonymous programmer under the Github handle 0v3y implemented iterative deepening depth-first alpha-beta search to solve 2048 [4]. Another user named Maarten Baert utilized a minimax-based AI to solve 2048, the source code of which is also availble on Github [5].

References

[1] `https://www.quora.com/What-is-the-expected-value-of-the-largest-tile-in-the-2048-game-if-you-randomly-press-the-arrows`

[2] `https://www.reddit.com/r/theydidthemath/comments/24ou9j request_what_is_the_lowest_score_needed_to_win/`

[3] `http://arxiv.org/abs/1312.5602`

[4] `https://github.com/ov3y/2048-AI`

[5] `https://github.com/MaartenBaert/2048-ai-emscripten`