CS221N Proposal: 2048 Challenge

Rui Qiu(ruiq), Yanpei Tian(yanpeit), Yulian Zhou (zhouyl)

October, 24, 2019

In this project, we aim to develop and compare different strategies for game 2048 (Figure 1). 2048 is a single player puzzle game, which is played on a 4×4 grid, with numbered tiles slide one step upon pressing 4 arrow keys. In the computer's turn, a new tile of value 2 will randomly appear in an empty grid. In the player's turn, the tiles will slide one step towards the direction of the pressed arrow key until it is blocked by other tiles or the edge of the grid. Upon colliding of the tiles with the same value, they will merge into a single tile with the value summing up. At each step, the score is won by the newly merged tiles' value, and the game ends when there is no legal moves (no empty grid and no merge can be performed).

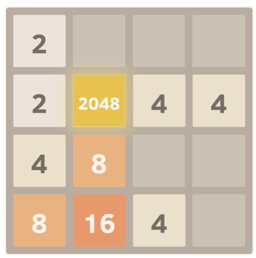

Figure 1: An illustrative grid view of a 2048 game in progress

The problem can be break down into several parts:

- IsEnd(s): Given a 4×4 grid, tell whether the game is over or not.
- Utility(s): The total score gained at the state s.
- Player(s): A human player who move the tiles on board, and a computer player who place a tile with value 2 at a random empty grid.
- Actions(s): A movement is valid for human player if the grid contains at least one empty cell so some tiles can slide towards the empty cell(s), or at any row or column, there is at least one adjacent pair so that they can collide upon move.
- Succ(s, a): A successor state of the board after taking an action a at a certain state s.

As shown above, 2048 can be reduced to a two-agent game, and given that the computer plays at random, we can define our evaluation function, and solve this problem with classic expectimax approach. Indeed, several studies have been carried out to address this problem with expectimax search[1, 2, 3]. Despite existence of several high performing models, we think this problem is a interesting topic. First, we want to explore different algorithms and compare their performance for the game. Second, we want to improve the computation efficiency, so that it can be calculated with high speed.

We plan to experiment mainly with the following two algorithms. 1. **Expectimax**. It achieves highest score according to our literature search, with the max tile valued 4096 or even higher. Our focus on this algorithm is to define good evaluation function (heuristics), as well as experimenting with different search depth. We will also apply alpha-beta pruning to promote the speed performance. 2. **Monte Carlo tree search**. It involves four state: selection, expansion, simulation and back-propagation. Though this method hasn't been explored very much for 2048, it is adapted by game Go, and defeated professional human players without handicaps. Additionally, we may try to integrate the scoring of expectimax into the search algorithm. If time permits, we would also like to explore some models with reinforcement learning, particularly deep reinforcement learning as we have a large state space.

As the game 2048 can play forever as long as there exists any valid move, we define two types of utilities for the end state – the total scores, and the total moves of a game. For evaluation, we will consider mainly the board layout and empty cell number. We will play around with the different metrics in all of the models.

For baseline, we have implemented two agents: a random agent, who moves completely at random directions as long as the move is valid, and a greedy agent, who moves towards the direction that leads to the highest score gain (Figure 2, Table 1). We noticed that the greedy agent performs better than the random agent in the trials. For oracle, we adapt the model by Navjinder[4], which uses deep reinforcement learning algorithm for training the AI. As the model runs very slow, we uses the data provided by the author (Table 1).

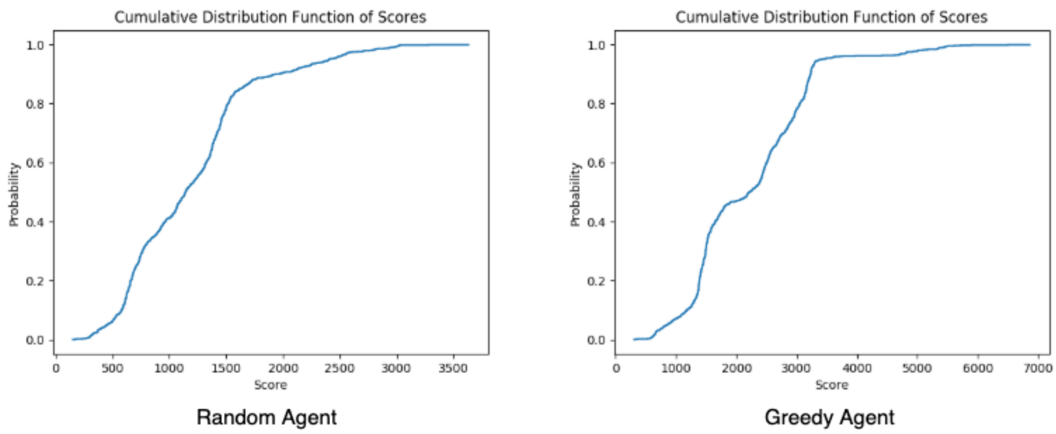

Figure 2: Cumulative distribution plots of the game scores by random agent and greedy agent (1000 games)

Table 1: Success rate out of 1000 games

Max tile score	Random Agent	Greedy Agent	Oracle (DRL)
2048	0%	0%	10%
1024	0%	0%	60%
512	0%	5%	95%
256	10%	50%	100%
128	60%	90%	100%
64	95%	100%	100%
32	100%	100%	100%

References

[1] Todd W Neller. Pedagogical possibilities for the 2048 puzzle game. *Journal of Computing Sciences in Colleges*, 30(3):38–46, 2015.

[2] Philip Rodgers and John Levine. An investigation into 2048 ai strategies. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–2. IEEE, 2014.

[3] Stackoverflow. What is the optimal algorithm for the game 2048? @ONLINE, 2014.

[4] Navjinder Virdee. Trained a neural network to play 2048 using deep-reinforcement learning @ONLINE, 2018.