RL Mini Project - 2048

Maadhav Gupta

March 2025

1 Introduction

> 2048 is a single-player sliding tile puzzle video game written by Italian web developer Gabriele Cirulli and published on GitHub. The objective of the game is to slide numbered tiles on a grid to combine them to create a tile with the number 2048; however, one can continue to play the game after reaching the goal, creating tiles with larger numbers. [WikiPedia]

The game is simple to define but difficult to solve. The action set is only {up, down, left, right}. However, the number of possible states is estimated to be 44 quadrillion ($\sim 10^{16}$). [jdleesmiller]

There are many algorithms that solve well for the game, with some algorithms being able to produce a 65536 tile for over 3% in terms of probability.

I try to look for simpler algorithms based on heuristics and patterns and incorporate them for a value function iteration and a Monte Carlo simulation over the future states.

2 Literature Review

Goenawan, Tao, Wu (2020) attempted an MCTS (Monte Carlo Tree Search) for the game. They considered the total sum of the tiles on the final board, the total 'merge' score (say, when 2 and 2 get combined to become 4, that 4 is added to the score) and the value of the biggest tile on the board for the value function. They discovered that the sum of the tiles on the final board ends up delivering the best results. On the other hand, when considering the value of the biggest tile, it leads to the worst results of the lot.

Nathaniel Thomas has a website for the game with an interactive UI where you can visually see the algorithms that he's used in play. One of the algorithms is the Monte Carlo Tree Search algorithm to choose the best move. It simulates many games from the current state, and chooses the move that leads to the highest average sum of tiles. It

is claimed that it wins around 75% of the time. He also uses N-Tuple Network trained using Temporal Difference Learning for over 3M+ games, which is said to should win almost 100% of the time, and it gets the 16384 tile around 60% of the time. Lastly, he also utilizes an Expectimax algorithm to search through all possible game states and choose the move with the highest likely score. This is claimed to achieve the 32768 tile around 30% of the time, however it is much slower than the other algorithms.

3 Dataset Source and Description

I have used this code to develop the logic of the game. The action set involved and the state spaces are simple to define. The code that I have referred to does this in an elegant manner and allows experimenting.

4 Experimentation

- Random Agent
  On multiple runs, it is observed the game is lost with the maximum tile score of 64 on average. The highest that I've observed over multiple trials has been a rare 256.

- Random Agent with foresight
  The random agent with foresight is an agent who at each state takes one particular action followed by a sequence of arbitrary actions. This can be for the next 100 actions or before termination. Then, the action which led to highest final sum of squares is chosen and is taken.
  This is does not do too much better than the random agent.

- Heuristic-based Random Agent with foresight
  In the algorithm, the value of each state is altered as per heuristics. There are certain strategies in the game to get a higher score.

  - Monotonicity
    Having the tiles in one row/column each pairwise being less than the tiles in some other row/column is desirable.
  - Corner high tile
    Keeping the highest tile(s) in the corner allows other tiles to get merged and does not waste space.
  - Higher number of empty tiles
    Having higher number of empty tiles at any state is, of course, desirable.

  Hence, positive weights are added to each of these factors for the states observed and then the random agent takes an action at each state. This is only a slight improvement over the other random agent variants.

- Permutational Heuristic
  Another traditional way to look at games with few and finite actions is considering the result with different permutations. Let's say {Left, down, left, down, left, down, ...} with the addition of a random move if {left, down} is not available. This incorporates the high values being confined near a corner and this is ensured which was difficult to implement in the previous version.
  The problem with this is that there is no learning in this as it is mostly determinsitic.

5 Conclusion

On many forums, it is debated that 2048 is very complex to solve with not very refined techniques. Even though intuitively, it should work, it takes a lot of games for an agent to optimally learn.

6 References

What's in a Game: Solving 2048 with Reinforcement Learning (Goenawan, Tao, Wu (2020))
The Mathematics of 2048: Counting States with Combinatorics
2048.ai (Nathaniel Thomas (2024))
poomstas