Learning to play the 2048 puzzle

Bakthavatsalam

Abstract

2048 is a popular sliding-tile puzzle played on mobile devices and web browsers. This paper discusses a strategy for an AI to solve the puzzle. The strategy is compared and evaluated with other methods.

INTRODUCTION

2048 is a sliding-block puzzle where the objective is to combine tiles by sliding and form the 2048 tile. The game can be played further to form larger numbers. Due to the stochastic nature of the game, a strategy must be developed that maximizes the average score and an optimal path to goal cannot be calculated [An Investigation into 2048 AI Strategies].

STRATEGIES FOR PLAYING GAMES

Minimax

The minimax algorithm is common for two player games where the strategy is as follows: choose a move to a position with the highest minimax value. The minimax algorithm can be formulated as shown [Optimal strategy in games with chance nodes]. The minimax algorithm is extended to alpha-beta pruning to reduce the number of nodes explored.

Expectimax

Expectimax is different from minimax in that chance nodes are interleaved in the search tree and decisions are based on the expected value of child nodes rather than the minimum or maximum values.

Monte Carlo methods

Monte-Carlo methods run many simulations of the game and use the average of these results in making a deciosn.

METHODOLOGY

In our method, we implement the pure Monte-Carlo game search combined with an evaluation function. The evaluation function is a weighted linear combination of selected board features. The weights for the function are learned using an optimization technique after several game sessions.

Pure Monte Carlo search

The evaluation function is used for the decision at each state. The procedure is shown in Algorithm 1:

Algorithm 1 Return action using evaluation function

```
1: procedure BESTACTION($s$)
2:     $score \leftarrow 0$
3:     $action \leftarrow NULL$
4:     for each a in ACTIONS(s) do
5:         $s' \leftarrow MOVE(s, a)$
6:         if $EVALUATE(s') > score$ then
7:             $score \leftarrow EVALUATE(s')$
8:             $action \leftarrow a$
9:     return $action$
```

The action at state 's' is the depth 1 search using the evaluation function. This strategy guides the simulations in the Monte-Carlo search. The results of the simulation are the number of moves survived by the AI (Algorithm 2).

Algorithm 2 Simulation Results

```
1: procedure SIM($s$)
2:     $score \leftarrow 0$
3:     while s not lose state do
4:         $action \leftarrow BESTACTION(s)$
5:         $s \leftarrow MOVE(s, action)$
6:         $score \leftarrow score + 1$
7:     return $score$
```

Finally, to determine the action at a state, the procedure in Algorithm 3 is used:

Evaluation function

Board features: To develop the evaluation function, few features from the game state were selected and the score was based on a linear combination of these features scaled by appropriate weights. These features are 1) the number of empty cells 2) the highest tile 3) the number of available moves 4) the sum of terms in the hadamard product of the board with a weight matrix. The last feature is to enforce the monotonic arrangement of tiles which is also the strategy employed by human players.

Algorithm 3 Return action from state after simulation

```
1:  procedure BESTACTIONSIM($s$)
2:      $score \leftarrow 0$
3:      $action \leftarrow NULL$
4:      for each a in ACTIONS(s) do
5:          $ActionScore \leftarrow 0$
6:          $s' \leftarrow MOVE(s, a)$
7:          for N iterations do
8:              $ActionScore \leftarrow ActionScore + SIM(s')$
9:          if $ActionScore > score$ then
10:             $score \leftarrow ActionScore$
11:             $action \leftarrow a$
12:     return $action$
```

64.0	32.0	16.0	8.0
0.5	1.0	2.0	4.0
0.25	0.125	0.0625	0.03125
0.001953125	0.00390625	0.0078125	0.015625

Table 1: Weight matrix to enforce monotonicity

Objective function For the purpose of optimization, an objective function must be defined. We have used the duration of a game session in terms of the number of moves made in the objective function. It is a measure of how long the AI is able to ”survive”. Due to the random nature of the game, the duration can vary significantly for the same set of parameters. Hence, the objective function is the average duration of ’N’ games played. For this project, we set N to 50.

Weight tuning Optimizing the weights for this is non-trivial as the search space is vast. As mentioned in the previous section, the objective function is also not deterministic for a set of parameters. Also, the derivatives for the function cannot be computed. For these reasons, a derivative free optimization technique has been used. A coarse estimate for a good set of candidate parameters was carried out using a grid search with a reasonable resolution. After a set of candidate parameters were found, a downhill-simplex algorithm with the starting point as the candidate solution was carried out. The results were used as the final weights.

RESULTS

The results for different values of the evaluation function, depth limits and search strategies are displayed on the table. With random moves as policy, the AI could only achieve a max tile of 512 and a duration of about 120 moves. An agent with a depth 1 search as the policy was able to achieve a 2048 tile 9 percent of the time and on average of about 330 moves, almost three times of that of random policy. It is clear that the optimization algorithm is improving performance. (Results for weights taken individually taken: expts to be done). A minimax search with a depth limit of 4 reached the 2048 tile about 40 percent of the time. It was not feasible to search deeper as it became computationally expensive and slow. (Experiments for the main monte carlo search needs to be carried out).

cell1	cell2	cell3	cell1	cell2	cell3
cell4	cell5	cell6	cell1	cell2	cell3
cell7	cell8	cell9	cell1	cell2	cell3

CONCLUSION

The average score and performance of the AI is ————- with the Monte Carlo search. Comparing the results of the optimized weights and weights taken individually, ——— ————. (Result about minimum and maximum tile : experiment to be done).