CandyCrush.ai: An AI Agent for Candy Crush

Jiwoo Lee, Niranjan Balachandar

October 30, 2016

1 Introduction

Candy Crush, a mobile puzzle game, has blown up in popularity in the past couple of years. The game looks deceptively simple, but it turns out to be a nontrivial task to "solve" Candy Crush. Though there are random elements to the game, there are valid strategies to employ in order to maximize the score. The state space is extremely large (on the order of 10^{50}, and getting an optimal score is an NP-hard problem [2]. In this project, we are implementing an AI agent that plays a slight variant (as defined in the Task Section) of Candy Crush with a goal of getting an optimal score.

2 Background

Candy Crush is a puzzle game based on a grid system. Each coordinate on the grid holds a candy, of which there are usually five types (usually distinguished by shape and color). Players earn points by swapping the positions of two candies to create rows or columns of 3 or more of the same candy. This is considered to be one move, and each move can obtain a score based on how many candies were connected in a row. Any move/swap that does not result in a row or column of 3 or more of the same candy does not count as a valid move. If there are no possible moves to be made, the game shuffles its board until a move is available. Once the candies are aligned in a row or column, they disappear, and all the candies above it are shifted down (new ones refill the grid from the top of the board). The game usually has a different end condition for each level, but we will focus on maximizing the score in a limited amount of moves.

3 Task Definition

3.1 Task

Our task in creating this Candy Crush agent is to have the AI be able to play the game and maximize the score. We will provide a randomly seeded grid layout and a number of turns for the AI to work with. Because we have to deal with the fact that we don't know what candies will fall from the top of the board after we clear candies, we will have some sort of Markov decision process we need to solve the optimal policy for.

Rules

Input: A a by b grid randomly initialized in each position with 1 of n different candies. For our task, we have set $a = b = 9$ and $n = 5$ to mimic the mobile Candy Crush application.

Actions: Let a turn be a valid swap of a pair of adjacent candies. A swap will be valid if it results in at least 3 of the same candies together in a row or column. The actions are a series of T turns to maximize total score. We have set $T = 50$.

Scoring: After each turn, a score will be awarded to the turn and all the scores from each turn will be summed into a total score. The objective is to maximize total score. After each turn t, the game will

iteratively scan the game grid for the biggest row/column of at least 3 of the same type candies, remove those candies, and drop down random candies to fill the grid, until there are no more rows/columns of at least 3 of the same type candies. In iteration number i, if the biggest group of connected candies in a row/column is of size c, where $c \geq 3$, then the score for turn t, iteration i will be:

$$\text{turnScore}_i(c) = (10c^2 - 10c)i$$

We have set this scoring function to mimic the scoring of the mobile Candy Crush applicatin. This scoring function rewards swaps that result in combos (when a swap results in multiple rows/columns of at least 3). For example, in a particular turn if there is a maximum group of 5 of the same type candies for the second iteration, then the turn score for that iteration is $(10 * 5^2 - 10 * 5)2 = 400$. The total score for a turn is the sum of the turn scores for all iterations i performed for that turn:

$$\text{turnScore}(t) = \sum_{\text{all } i} \text{turnScore}_i$$

The total score for the game is the sum of the turn scores for all turns:

$$\text{totalScore} = \sum_{t=1}^{T} \text{turnScore}(t)$$

The objective of the game is to maximize the total score.

Output: Total Score

3.2 Evaluation

We will evaluate the success of the AI by finding the average score of multiple trials with randomly seeded grids and candies. We will compare how the average score compares to that of the oracle (explained later).

3.3 Infrastructure

Our Candy Crush simulator uses a 9 by 9 2d numpy array to create the grid on which we will play the game. The different candies are represented by different integers in the 2d array. When the human is playing, it will take in two coordinates, and if the swap is valid (creates a row or column of 3 or more pieces), the swap will be executed and the resulting board is shown.

4 Approach

We will try to model this AI algorithm as a Markov decision process. When we make a move, we know the immediate reward (score that eliminating those candies gives us) but we also need to consider the resulting state, which will most likely consist of the board and the potential candies that could fall to fill the holes left by the move. Because the state space is enormous, we will not implement a naive QLearning algorithm; instead, we will create a feature vector to augment the algorithm's ability to adapt to states it hasn't seen before, which will be the majority of states.

There are many challenges that come with trying to solve a game with this huge a state space. Time complexity is a huge issue that comes with solving a grid based problem like this, because we need to check for new candies in a row/column every move we make. When we add in new candies after deleting the candies in the original row/column, we must again make sure that the candies in a row are deleted. Checking for this in each iteration in a 9 by 9 is quite exhaustive, but we hope to optimize these checks eventually.

4.1 Baseline

Our baseline AI iterates through the rows and columns of the game grid until it finds the first possible valid move and swap those candies. The baseline AI doesn't take into account the orientation of the rest of the candies on the board, and will take the first swap it sees. The average AI score over a 10,000 trials was 8375.

4.2 Oracle

Our oracle is Jiwoo Lee, an experienced Candy Crush player with over 3 years of swiping under his belt. The average oracle score over 10 trials was 31714.

5 Literature Review

There are a couple Candy Crush python bots online, but one is quite well documented. The Candy Crush Bot by Alexandru Ene is written in python, and implements a greedy-like algorithm that performs the highest scoring move on the board at each turn. While our algorithm will definitely take into consideration the highest scoring move on the board, we hope to be able to extrapolate more from the other candies on the board to take into account resulting combos to make the most optimal move [2]. Ene's algorithm fails to consider anything besides the immediately relevant candies, so hopefully we can make a notable difference there while being complementary to his approach.

6 Sample Inputs and Outputs

```
Turns left: 50
SCORE 0
[[3 4 1 4 2 4 2 5 1]
 [1 4 3 2 5 4 4 5 3]
 [4 2 5 5 3 1 5 4 3]
 [2 1 1 2 2 1 2 2 5]
 [1 2 3 3 1 3 2 1 1]
 [2 4 2 5 2 2 1 3 2]
 [5 1 2 2 3 5 5 1 1]
 [2 1 5 1 2 5 2 3 3]
 [3 4 4 3 5 1 3 5 1]]
```

Sample AI Baseline Input

```
Turns left: 0
[[4 3 3 2 4 2 3 4 5]
 [4 4 1 5 1 3 1 5 2]
 [1 3 1 4 5 3 1 2 3]
 [1 1 3 1 4 5 2 3 1]
 [5 4 4 2 1 5 1 4 5]
 [2 3 4 2 1 2 4 5 4]
 [3 5 3 1 5 3 1 1 5]
 [5 3 1 2 1 1 4 1 5]
 [1 3 4 4 3 1 1 4 4]]
1 FINAL SCORE 7700
```

Sample AI Baseline Output

```
Turns left: 50
SCORE 0
[[2 3 5 4 3 5 1 1 2]
 [1 2 4 2 2 3 3 1 3]
 [1 4 1 4 3 4 2 3 1]
 [3 4 1 1 2 1 4 2 1]
 [3 5 4 3 1 4 5 1 3]
 [2 4 3 5 1 2 4 2 3]
 [1 2 4 2 4 1 3 2 2]
 [3 1 2 4 5 1 5 3 1]
 [2 5 4 2 1 3 1 5 4]]
Enter first coordinate to switch in the form "row,col": 7,2
Enter second coordinate to switch in the form "row,col": 7,3
```

Sample Oracle (Human) Input

```
Turns left: 0
[[1 3 5 5 3 4 4 3 3]
 [4 5 2 1 3 5 1 5 5]
 [3 1 4 2 4 1 5 3 1]
 [5 2 5 4 2 2 5 3 2]
 [4 3 2 4 3 5 2 1 1]
 [1 1 5 3 1 3 4 4 3]
 [3 3 1 5 3 4 2 4 3]
 [4 2 3 2 5 3 2 3 2]
 [3 1 4 4 2 2 1 5 4]]
FINAL SCORE 41680
```

Sample Oracle (Human) Output

References

[1] https://arxiv.org/pdf/1403.1911v1.pdf
[2] https://github.com/AlexEne/CCrush-Bot