Analysis of game Yachtee through reinforcement learning

Minsuk Chang

April 26, 2023

Abstract

In this paper, I have trained an AI based on reinforcement learning and Deep Q Networks, which achieved an average score of almost 190. By utilizing this human-level performing model, Yachtzee is analyzed in detail, including the probability model for rolling dice in certain circumstances. Furthermore, the possible improvements for the model architectures are shown in the summary of this paper.

1 Introduction

Yahtzee is a simple dice game played with five six-sided dice. The player's role is to roll the dice for limited times and fill in(lock) the corresponding score on the table. The system of probabilities controls the game so that it can be represented by Markov Decision Process(MDP). This enables designing and training an agent by reinforcement learning. This research will analyze the game and find the optimal policy to maximize its score by comparing statistical and RL-related methods.

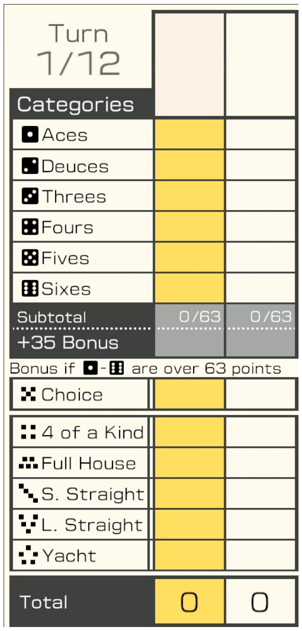
2 Rules of Yacht

2.1 Rolling Dice

The most important thing while playing Yacht is to roll dice to make target pairs. Each time every player has three opportunities to roll the dice. After initial rolling, one can hold any amount of dice they want and keep them from being rolled. By keeping them, players can increase their chances of making desirable pairs. Players must constantly think of probability and choose which dices to keep or roll to make higher-ranked patterns of dices. After rolling the dice three times which is the maximum limit, the player must fill in the scoreboard before he can roll the dice again three times.

2.2 The Score Board

Every time the player rolls his dice, he can stop rolling and fill in the scoreboard according to the pattern in his dice. The board has two main sections: Number columns(Aces - Sixes) and Special columns(Choice - Yacht). Each column can be locked only once throughout the game, so to maximize the overall score, players have to show their strategy to choose which columns to fill in first.
The score calculated for the number section is a simple sum of corresponding dice. For example, if you have three sixes and two ones in your dice, you can record 1*2=2 points in Aces or 6*3=18 points in Sixes. Which column to lock is the player's decision. If the sum of scores in the number section exceeds 63, then the player receives additional 35 points for the overall score.
For the Special Section, the dice must match the column's specific pattern or only a point of 0 can be locked. The details for the score corresponding to each column can be found in figure2.



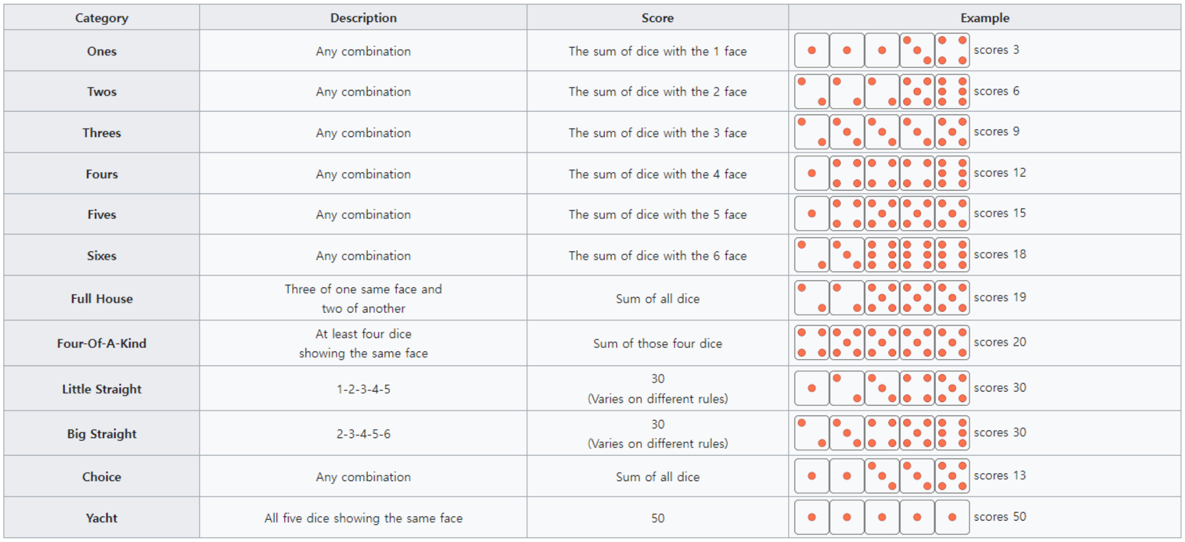
Turn
1/12

Categories		
⚀ Aces		
⚁ Deuces		
⚂ Threes		
⚃ Fours		
⚄ Fives		
⚅ Sixes		
Subtotal	0/63	0/63
+35 Bonus		

Bonus if ⚀-⚅ are over 63 points

Choice		
4 of a Kind		
Full House		
S. Straight		
L. Straight		
Yacht		
Total	0	0

Figure 1: An example of Yacht scoreboard for 2 players.



Category	Description	Score	Example
Ones	Any combination	The sum of dice with the 1 face	⚀⚀⚀⚂⚃ scores 3
Twos	Any combination	The sum of dice with the 2 face	⚁⚁⚁⚄⚅ scores 6
Threes	Any combination	The sum of dice with the 3 face	⚁⚂⚂⚂⚄ scores 9
Fours	Any combination	The sum of dice with the 4 face	⚀⚃⚃⚃⚅ scores 12
Fives	Any combination	The sum of dice with the 5 face	⚀⚃⚄⚄⚄ scores 15
Sixes	Any combination	The sum of dice with the 6 face	⚁⚂⚅⚅⚅ scores 18
Full House	Three of one same face and two of another	Sum of all dice	⚁⚁⚄⚄⚄ scores 19
Four-Of-A-Kind	At least four dice showing the same face	Sum of those four dice	⚃⚄⚄⚄⚄ scores 20
Little Straight	1-2-3-4-5	30 (Varies on different rules)	⚀⚁⚂⚃⚄ scores 30
Big Straight	2-3-4-5-6	30 (Varies on different rules)	⚁⚂⚃⚄⚅ scores 30
Choice	Any combination	Sum of all dice	⚀⚀⚂⚂⚄ scores 13
Yacht	All five dice showing the same face	50	⚀⚀⚀⚀⚀ scores 50

Figure 2: Dice patterns and corresponding scores.

2.3 Game Over

There are 12 columns to lock on the scoreboard, and this means after all of the columns and sections are filled, then the game finishes. To reach this, the player can choose to keep and roll his dice two times per column(an initial roll is necessary).

3 Yacht for Reinforcement Learning

3.1 States

The state of the Yacht includes several information about the game's status. The most important feature is the dices and the scoreboard. Also, the state must include information about the number of chances left to roll the dice and the partial sum of the Number section on the scoreboard to calculate the bonus score.

To pack all this information into single state representation, I've created a 1D tensor which lengths 19. The first 12 numbers represent the status of each column on the board. The number means the scores the player can get if he fills in that column. If the column is locked, then the number is locked to 0.

The 13th number represents the partial sum of the number section of the scoreboard. If the number exceeds 63, then the player receives a bonus score.

The 14th number represents the number of chances left for rolling the dice. The initial roll is done automatically, so the number only can be in the range of 0, 1, 2.

The 15th to 19th numbers represent the current status of the dice.

These numbers are divided with constant numbers to keep those numbers between 0 and 1.

3.2 Actions

There are a maximum of 44 actions available for every state. The first 32 actions correspond to rolling dice. Each number represents which dice to keep and which dice to roll. For example, action 11 corresponds to (0,1,0,1,1) in binary. This means that 2,4,5th dice are fixed while others are rolled randomly.

The next 12 actions correspond to the columns of the scoreboard to fill in the score. The number of columns follows the order of categories in Figure 1. For example, action 38 means filling in 38-32=6th columns of scoreboard which is Sixes.

However, some actions may be restricted while the scoreboard is getting filled and the dice are rolled. For example, once the score on column Aces is filled, action 33(fill in Aces) cannot be chosen until the current episode ends. Similarly, if the dice are rolled two times in a row, there are no more chances to roll the dice. In this case, all options for rolling dice (actions 1 to 32) are prohibited until one of the empty columns is filled.

3.3 Network

The network consists of only Linear layers, also known as fully connected layers. The original paper of DQN, which plays Atari games, has various convolution layers with batch normalization. Still, in the current network, input states are 1d-tensor, each representing different aspects of the single game, leading to a simple structure. The number of parameters used in our network is about 30 million.

4 Strategies

4.1 State Representation

In the initial trials, simple binary states represented the board, such as 1 for locked and 0 for not locked. Humans can easily calculate the scores likely to be locked on each column with the current dice. However, it was found that for machines, quite deep layers are required for the network to capture such high-dimensional information.

To handle this, I've changed each binary feature to continuous ones, representing scores the player can obtain when locking that column.

Additionally, to reduce the number of states representing the same pattern of dice, the game was implemented to automatically sort the array of dice after each time the dice is rolled.

4.2 Network architecture

The network is composed of simple linear layers. Initial trials which used three stacked layers were poorly performed. I think this happened because each number composing the state has different meanings. To differentiate these parts, the state was split into four categories. Each category represents the number section, special section, rolled dice, and rolls left. The numbers are separated and sent to the corresponding layer, which only extracts information about particular categories. After the base feature is removed, all features are passed through the last layer to mix those features and create new ones. The layers, except the initial layer, resemble the shape of an encoder network, making the network generate highly condensed information used for precisely approximating the Q-values.

4.3 Value Normalization

State values may range differently depending on the categories they represent. For example, numbers in a particular category range from -1 to 65, while dice only range from 1 to 6. Also, rewards may range from 0 to 65 because players can get at most 65 points in a single action(five 6 in dice and write on Sixes with bonus(30+35=65)). To make learning more stabled, I made all numbers in the states between 0 and 1, except -1, representing that the board's row is locked.

4.4 Action Repeat

The Yacht is entirely run by probability, and the most crucial point is to learn the state transition and generalize those results. Complete information needed to generalize one single state-action pair is about $6^5 = 7776$ times of trial. This means that a single step needs to be reappeared in the later episodes more than 7000 times to generalize, which is not likely to happen easily. To manage this, I made each step in the episode repeated over a hundred times to have sufficient trials for the network to reduce average loss and easily generalize.

5 Training & Results

5.1 Environment

The training is done in the Google colab-pro environment with P100 GPU. The real-time training status is observed and logged by the tensorboard.

5.2 Parameters

Target Network Update: once per 50 episodes.
Action Repeat: 128 times per step.
Epsilon Decay : decay following formula: $\epsilon = end + (start - end) * e^{-(episode/decay)}$
(start = 1, end = 0.1, decay=5000)
Learning Rate: manually reduced by episode.

5.3 Obtained Scores

After the Train was saturated, the average score of 11000 runs stayed around 185, around the average score obtained by humans. Also, The score ranged widely due to the game's characteristic, which is based on pure statistics.

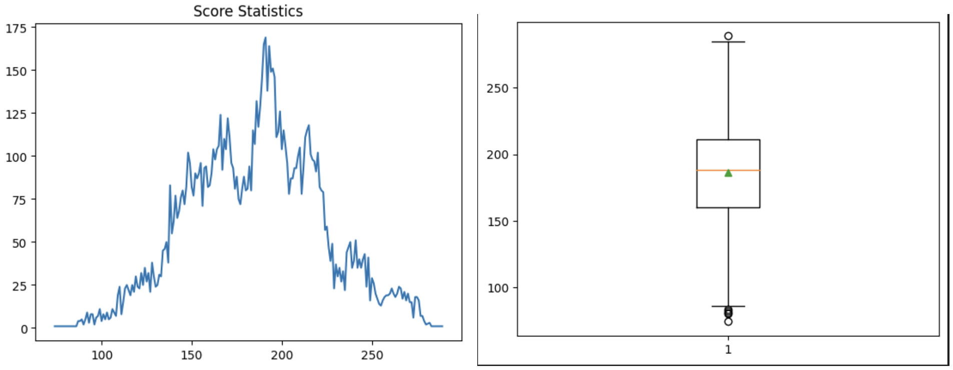


Figure 3: Statistics of the overall score for 11000 trials

6 Analysis

6.1 Statistic Analysis

Among the 11000 trials of the final model, each case corresponding to the status of dice is grouped and aggregated to show the most and second most popular choices for every dice status. Each circumstance is represented as a tuple consisting of (Action, Proportion) such as (Lock Full House, 64

6.2 Yacht

The primary goal of this game is to maximize the overall score locked on the scoreboard. To achieve this, when there are no dice left to roll, we should select the scoreboard row which would be the most difficult to produce such a pattern. For example, if there were five 1s, it would be appropriate to lock Yacht because it has the largest score and the smallest chance of happening. However, the average percentage of selecting Yacht for the same five numbers on the dice was around 70%, and the other 20% was locking the corresponding number section. This may be reasonable for large dice such as 5 or 6 because locking the number section corresponding to those large dice would potentially lead to a bonus score of +35. Therefore, the percentage for locking Yacht was above 90% for dice 1 and around 70% for dice 6. This example shows the importance of bonus points for the overall score and the impact of large dice numbers.

6.3 Full House

The most interesting part of this game is the full house and the four-of-a-kind. The dice sum purely determines the allocated score for locking these two rows. This fact creates the decision to roll again even though the pattern is already made. For example, if the dice is [2, 2, 1, 1, 1], which would give a score of 7 for locking the full house, additional rolling might generate a more valuable pattern such as (1, 1, 1, 1, 1) and lead to an instant reward of 50 points. Even though the AI chose to lock the scoreboard, it decided to lock the ones but not the full house. It means that the minimum score suitable for a full house is much larger. Through the experiment, I could find out that the minimum scored pattern to lock a full house was [5, 5, 5, 1, 1] and [5, 5, 1, 1, 1], when 5 was the majority, and the minority of the dice respectively.

6.4 Small & Large Straights

One of the patterns that is hard to decide is [6, 5, 4, 2, 2]. It's because the only likable pattern it could create in a single roll of dice is definitely [6, 5, 4, 3, 2], and the next likable one is [6, 5, 4, 3, X]. In calculation, the chance of getting the large straight by rolling one of those 2s is 1/6. However, there is a strategy of rolling two 2s together. This has two kinds of benefits. First, there is an insurance that it may create [6, 5, 4, 3, X] for a probability of 11/36. Also, it may create a large straight with a probability of 2/36. The second one has less probability of making a large straight but is much safer because it might create a small straight in a relatively big likelihood.

7 Summary

Even though the model achieved a relatively high overall score for the game, it is not perfect regarding action selections. Because the game Yacht is based purely on probability, the model's training was quite unstable even though the average of 100 runs measures the performance. This led to the result shown as the final analysis of 11000 runs. The model didn't choose the most appropriate action for obvious patterns such as [1, 1, 1, 1, 1], which would be locking the Yacht. Therefore, to mitigate the model's unstableness, network architecture, and the training method must be changed to handle various probabilistic scenarios.

However, it was interesting that the model could differentiate the case of dice which has the same pattern but different reward, such as [6, 6, 6, 6, 1] and [1, 1, 1, 1, 2], by selecting different actions. Moreover, for the case of large and small straights, the action chosen by the model matched the appropriate action deduced by the statistical analysis.

References