

Analysis of game Yacht through reinforcement learning

Minsuk Chang

January 26, 2023

Abstract

Your abstract.

1 Introduction

Yacht is a simple dice game played with 5 of six-sided dices. Player's role is to roll the dice for limited times and fill in the corresponding score on the table. The game is controlled by the system of probabilities, so it can be represented by Markov Decision Process(MDP). This enables designing and training an agent by reinforcement learning. In this research, we 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 in order to make target pairs. Each time every players have 3 changes to roll the dice. After initial rolling, one can hold any amount of dice they want and keep them from being rolled. By this action of keeping, players can increase their chances to make desirable pairs. Players have to constantly think of probability and choose which dices to keep or roll to make higher ranked patterns of dices. After rolling dices for 3 times which is the maximum limit, player must fill in the scoreboard before he can roll the dices again for 3 times.

2.2 The Score Board

Every time the player rolls his dice, he can stop rolling and fill in the score board according to the pattern in his dices. There are 2 main sections on the board which are Number columns(Aces - Sixes) and Special columns(Choice - Yacht). Each columns can be written only once throughout the game, so in order to maximize the overall score, players have to show their strategy to choose which columns to fill in first.

The score calculated for number section is simple sum of corresponding dices. For example, if you have 3 sixes and 2 ones in your dices, then you can either record $1*2=2$ points in Aces or $6*3=18$ points in Sixes. Which column to write is the player's decision. If the sum of scores in the number section exceeds 63, then player receives additional 35 points for the overall score.

For the Special Section, the dices must match the column's pattern or only 0 points can be written. The details for the score corresponding to each columns can be found on figure2.

2.3 Game Over

There are total 12 columns to write on the score board and this means after all of the columns and sections are filled, then the game finishes. To reach this, player can choose to keep and roll his dices for 2 times per column(initial roll is necessary).

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.

3 Yacht for Reinforcement Learning

3.1 States

The state of yacht includes several information about the game's status. The most important feature is the dices and the score board. Also, state needs to include information about the number of chances left to roll the dices, and the partial sum of Number section on the score board in order to calculate the bonus score.

In order to pack all these information into single state representation, I've created 1D-tensor which lengths 19. First 12 numbers represent status of each column on the board. Number itself means that player can get that amount of score if he fill in that column. If the column is fixed, then the number is fixed to 0.

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

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 range of 0, 1, 2.

15th to 19th numbers represent the current status of the dices.

All of these numbers are divided with constant numbers in order to keep those numbers in range between 0 and 1.

3.2 Actions

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

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

However, while the scoreboard is getting filled and the dices are rolled, some actions may be restricted. For example, once the score on columns Aces are filled, then action 33(fill in Aces) cannot be chosen until current episode ends. Similarly, if the dices are rolled for 2 times in a row, there are no more chances to roll the dice. In this case, all options for rolling dices (action 1 to 32) is prohibited until one of the empty columns are filled.

3.3 Network

The network consists of only Linear layers which are also known as fully connected layers. Original paper of DQN which plays Atari games has various convolution layers with batch normalization, but in current network, input states are 1d-tensor and each represents different aspects of single game which led to 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 state were used for representing board, such as 1 for fixed and 0 for not fixed. Humans can easily calculate the scores which is likely to be written on each column with current dices. However, it was found that for machines, quite deep layers are required for the network in order to capture such high-dimensional information.

To handle this, I've decided to change each binary features to continuous ones which can actually represent scores which the player can obtain when the he chooses to fill in that column.

Additionally, in order to reduce the number of states which represent same pattern of dices, the game was implemented to automatically sort the array of dices after each time the dice is rolled.

4.2 Network architecture

Network is composed of simple linear layers. Initial trials which used 3 layers which is simply stacked performed badly. I think this happened because each numbers composing the state has different meanings. To separate these parts of the state, state was split into 4 categories. Each categories represent number section, special section, dices and dices left. The numbers are separated and sent to the corresponding layer which only extracts information related to particular categories. After the base feature is extracted, all features are passed all the way through the last layer in order to mix those features and create new ones. The layers except initial layer resembles the shape of an encoder network, which makes the network to generate highly condensed information which is used for precisely approximating the Q-values.

4.3 Value Normalization

State values may range differently depending on the categories which it represents. For example, numbers in special category would range from -1 to 65, while dices would range from 1 to 6 only. Also, rewards may range from 0 to 65 too, because players can get at most 65 points in single action(five 6 in dices and write on Sixes with bonus(30+35=65)). To make learning more stabled, I made all numbers in the states to be between 0 and 1, except -1 which represents that the board is fixed.

4.4 Action Repeat

Yacht is completely run by probability and the most important point is to learn the state transition and generalize those results. Full information needed to generalize one single state-action pair is about $6^5 = 7776$ times of trial. This means that 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 to be repeated over hundred times to have sufficient trials for the network to reduce averaged loss and easily generalize.

5 Training

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 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.

References