Learning Yahtzee Using Deep Reinforcement Learning

Nick Pape
nap626
nickpape@utexas.edu

Abstract

Write your abstract here. This should be a concise summary of your work, including the problem you're addressing, your approach, and key results. Keep it to about 150-250 words.

1 Introduction and Research Background

At first glance, *Yahtzee* appears to be a simple dice game. Simply roll some dice, choose which to keep, and score points based on the resulting combinations. While arithmetic may be the minimum skill required to play, playing the game well requires strategic decision-making, probabilistic reasoning, and risk management.

In this paper, we explore the application of various reinforcement learning techniques to optimize gameplay in Yahtzee. We investigate how different reinforcement learning algorithms, model architectures, state and action representations, and reward structures impact the performance of agents in this complex environment.

1.1 Background

1.1.1 Rules of Yahtzee

Yahtzee is played with five standard six-sided dice and a shared scorecard containing 13 categories. Turns are rotated among players. A turn starts with a player rolling all five dice. They may then choose to keep some dice, and re-roll the remaining ones. This process can be repeated up to two more times, for a total of three rolls. After the final roll, the player must select one of the 13 scoring categories to apply to their current dice. Each category has specific scoring rules, and each can only be used once per game.

1.1.2 Mathematical Representation of Yahtzee

The space of all possible dice configurations is:

$$\mathcal{D} \in \{1, 2, 3, 4, 5, 6\}^5$$

and the current state of the dice is represented as:

$$\mathbf{d} \in \mathcal{D}$$

In addition, we can represent the score card as a vector of length 13, where each element corresponds to a scoring category:

$$\mathbf{c} = (c_1, c_2, \ldots, c_{13}) \text{ where } c_i \in \mathcal{D}_i \cup \{\varnothing\}$$

and $\varnothing$ indicates an unused category.

Let us also define a dice face counting function which we can use to simplify score calculations:

$$n_v(\mathbf{d}) = \sum_{i=1}^{5} \mathbb{I}(d_i = v), \quad v \in \{1, \ldots, 6\}$$
$$\mathbf{n}(\mathbf{d}) = \big(n_1(\mathbf{d}), \ldots, n_6(\mathbf{d})\big)$$

Let the potential score for each category be defined as follows (where detailed scoring rules can be found in Appendix):

$$\mathbf{f}(\mathbf{d}) = \big(f_1(\mathbf{d}), f_2(\mathbf{d}), \ldots, f_{13}(\mathbf{d})\big)$$

The domain of possible scores in a score card for each category is then:

$$\begin{aligned}
\mathcal{C}_i &= \Big\{ i \cdot k : k \in \{0, 1, \ldots, 5\} \Big\} \forall i \in \{1, \ldots, 6\} \\
\mathcal{C}_7 &= \mathcal{C}_8 = \mathcal{C}_9 \\
&= \{0, 1, 2, \ldots, 30\} \\
\mathcal{C}_9 &= \{0, 25\} \\
\mathcal{C}_{10} &= \{0, 30\} \\
\mathcal{C}_{11} &= \{0, 40\} \\
\mathcal{C}_{12} &= \{0, 50\}
\end{aligned}$$

The current turn number can be represented as:

$$t \in \{1, 2, \ldots, 13\}, \quad t = \sum_{i=1}^{13} \mathbb{I}(c_i \neq \varnothing)$$

A single turn is composed of an initial dice roll, two optional re-rolls, and a final scoring decision. Let $r = 0$, with $r \in \{0, 1, 2\}$ which is the number of rolls taken so far. Prior to the first roll, the dice are randomized:

$$\mathbf{d}_{r=0} \sim U(\mathcal{D})$$

The player must decide which dice to keep and which to re-roll. Let the player define a keep vector:

$$\mathbf{k} \in \{0, 1\}^5$$

where $\mathbf{k}_i = 1$ indicates that die i is kept, otherwise it is re-rolled.

We can then define the transition of the dice state after a re-roll as:

$$\begin{aligned} \mathbf{d}' &\sim U(\mathcal{D}), \\ \mathbf{d}_{r+1} &= (\mathbf{1} - \mathbf{k}) \odot \mathbf{d}' + \mathbf{k} \odot \mathbf{d} \end{aligned}$$

When $r = 2$, the player must choose a scoring category to apply their current dice to. Define a scoring choice mask as a one-hot vector:

$$\mathbf{s} \in \{0, 1\}^{13}, \quad \|\mathbf{s}|_1 = 1$$

For the purposes of calculating the final (or current) score, any field that has not been scored yet can be counted as zero. We can define a mask vector for this:

$$\begin{aligned} &\mathbf{u}(\mathbf{c}) \in \{0, 1\}^{13} \\ &\mathbf{u}(\mathbf{c})_i = \mathbb{I}\big(c_i \neq \varnothing\big), \quad \forall i = \{1, \ldots 13\} \end{aligned}$$

If a player achieves a total score of 63 or more in the upper section (categories 1-6), they receive a bonus of 35 points:

$$B(\mathbf{c}) = \begin{cases} 0, & 63 \leq \sum_{i=1}^{6} \mathbf{u}(\mathbf{c})_i \cdot \mathbf{c}_i \\ 35, & \text{otherwise} \end{cases}$$

The player's score can thus be calculated as:

$$S(\mathbf{c}) = B(\mathbf{c}) + \big\langle \mathbf{u}(\mathbf{c}), \mathbf{c} \big\rangle$$

1.2 Complexity Analysis

1.2.1 State Space Analysis

The state space for a solitaire game of Yahtzee can be described by the tuple of the current dice configuration, the scorecard status, and the roll count within the turn:

$$\mathbf{S} = (\mathbf{d}, \mathbf{c}, r)$$

The cardinality of the state space is primarily driven by the score card, $\mathcal{C}$:

$$|\mathcal{C}| = 7^6 \cdot 32^3 \cdot 3^4 \approx 3.12 \times 10^{11}$$

However the dice configuration space is also significant:

$$|\mathcal{D}| = 6^5 = 7776$$

And the roll count contributes a factor of 3:

$$|r \in \{0, 1, 2\}| = 3$$

Thus the state space has a cardinality of:

$$\begin{aligned} |\mathbf{S}| &= |\mathcal{D}| \cdot |\mathcal{C}| \cdot |r| \\ &= 6^5 \cdot 7^6 \cdot 32^3 \cdot 3^4 \cdot 3 \\ &\approx 7.28 \times 10^{15} \end{aligned}$$

1.2.2 Infeasibility of Tabular Methods

If this space were to be solved with tabular dynamic programming techniques using 16-bit half-precision floats, it would require nearly 1.5 petabytes of memory to store. While there have been prior attempts to use tabular methods for Yahtzee, these required significant optimizations and clustered computing resources to solve directly (**?**).

For a local-play setting, this makes traditional tabular methods infeasible for Yahtzee, necessitating the use of function approximation methods such as deep neural networks.

1.2.3 Neural Network Size Bounds

One challenge in applying neural networks to approximate the value or policy function for Yahtzee is in determining an appropriate network size. We will quickly consider theoretical upper and lower bounds for the number of neurons required using both the full state space and a reduced representation.

Assume for simplicity our reduced representation consists of the dice state $\mathbf{d}$ encoded one-hot, the roll count r encoded one-hot, the bin count

vector $\mathbf{n}(\mathbf{d})$ encoded one-hot, and a score mask $\mathbf{u}(\mathbf{c})$. This gives input dimensionality of $30 + 3 + 30 + 13 = 52$ and cardinality of $2^{52} \approx 4.5 \times 10^{15}$.

For a discrete control problem with a finite state size of $|\mathbf{S}|$, there are classical results (Horne and Hush, 1994) showing that a recurrent neural network can represent any $|\mathbf{S}|$-state finite state machine with a number of neurons that grows only logarithmically with $|\mathbf{S}|$. This gives us an upper bound of $O(\sqrt{|\mathbf{S}|}) \approx 8.4 \times 10^7$ neurons. However, this is a prohibitively large number of neurons for practical training and inference.

For a lower bound, Hanin (2017) shows any continuous, convex function can in principle be approximated by a network whose hidden layer width has a lower bound of $\Omega(d + 1)$, and a non-convex function can be approximated by a network with a hidden layer width of at least $\Omega(d+3)$, where d is the input dimension. For a example, assume Yahtzee is non-convex, and we represent the dice state $\mathbf{d}$ and r as one-hot encodings and utilize the score mask $\mathbf{u}(\mathbf{c})$, then $x = 6 \cdot 5 + 3 + 13 = 46$ means our *minimum* hidden layer width is 49 neurons.

In practice, the ideal network size is not driven by the cardinality of the state space, but rather by the

1.3 Intra-turn Decision Complexity

Test.

1.4 Problem Statement

Clearly state the problem you are addressing.

1.5 Research Objectives

Outline your research objectives and what you aim to achieve.

2 Research and Methods

Describe your approach and methods here.

2.1 Methodology Overview

Provide an overview of your research methodology and approach.

2.2 Model Architecture

Describe your model or approach in detail.

2.3 Experimental Design

Describe your experimental design and evaluation metrics.

3 Materials and Data Sources

Describe the dataset and materials you are using.

3.1 Dataset Description

Provide detailed information about your dataset, including size, characteristics, and source.

3.2 Data Collection and Preprocessing

Explain how data was collected and any preprocessing steps performed.

3.3 Tools and Technologies

List the tools, frameworks, and technologies used in your research.

4 Results

Present your experimental results here.

Method	Metric 1	Metric 2
Baseline	XX.XX	XX.XX
Your Method	XX.XX	XX.XX

Table 1: Results comparison

5 Discussion and Conclusion

5.1 Discussion of Results

Discuss your findings, their implications, and how they relate to existing literature.

5.2 Limitations

Acknowledge the limitations of your study and approach.

5.3 Future Work

Discuss potential directions for future research.

5.4 Conclusion

Summarize your work and its contributions to the field.

6 AI Usage Disclosure

This paper utilized artificial intelligence tools in the following ways:

- **GitHub Copilot** was used for initial typesetting assistance with the LaTeX document structure and formatting.

- **ChatGPT GPT-5** was used for brainstorming ideas for reinforcement learning applications in games, helping to narrow down to a particular game that was both non-trivial and interesting for this research.

All other content, including research methodology, analysis, results interpretation, and conclusions, represents original work by the author. The AI tools were used only for initial ideation and technical formatting support, not for generating substantive content or analysis.

GitHub Copilot was also used to write this section.

7 Acknowledgments

Acknowledge any help or resources you used.

References

Boris Hanin. 2017. Universal function approximation by deep neural nets with bounded width and relu activations.

Bill G. Horne and Don R. Hush. 1994. Bounds on the complexity of recurrent neural network implementations of finite state machines. In *Advances in Neural Information Processing Systems 6*, pages 359–366, San Francisco, CA. Morgan Kaufmann.

A Full Yahtzee Scoring Rules

Next we define the indicator functions for each of the scoring categories:

$$\mathbb{I}_{\text{3k}}(\mathbf{d}) = \mathbb{I}\big\{\max_v n_v(\mathbf{d}) \geq 3\big\}$$

$$\mathbb{I}_{\text{4k}}(\mathbf{d}) = \mathbb{I}\big\{\max_v n_v(\mathbf{d}) \geq 4\big\}$$

$$\mathbb{I}_{\text{full}}(\mathbf{d}) = \mathbb{I}\Big\{\exists i, j \in \{1,\ldots,6\} \text{ with } n_i(\mathbf{d}) = 3 \wedge n_j(\mathbf{d}) = 2\Big\}$$

$$\mathbb{I}_{\text{ss}}(\mathbf{d}) = \mathbb{I}\Big\{\exists k \in \{1,2,3\} \text{ with } \sum_{v=k}^{k+3} \mathbb{I}\{n_v(\mathbf{d}) > 0\} = 4\Big\}$$

$$\mathbb{I}_{\text{ls}}(\mathbf{d}) = \mathbb{I}\Big\{\exists k \in \{1,2\} \text{ with } \sum_{v=k}^{k+4} \mathbb{I}\{n_v(\mathbf{d}) > 0\} = 5\Big\}$$

$$\mathbb{I}_{\text{yahtzee}}(\mathbf{d}) = \mathbb{I}\big\{\max_v n_v(\mathbf{d}) = 5\big\}$$

The potential score for each category can then be defined as:

$$f_j(\mathbf{d}) = j \cdot n_j(\mathbf{d}), \qquad j \in \{1,\ldots,6\}$$

$$f_7(\mathbf{d}) = \mathbf{1}^\top \mathbf{d} \cdot \mathbb{I}_{\text{3k}}(\mathbf{d})$$

$$f_8(\mathbf{d}) = \mathbf{1}^\top \mathbf{d} \cdot \mathbb{I}_{\text{4k}}(\mathbf{d})$$

$$f_9(\mathbf{d}) = 25 \cdot \mathbb{I}_{\text{full}}(\mathbf{d})$$

$$f_{10}(\mathbf{d}) = 30 \cdot \mathbb{I}_{\text{ss}}(\mathbf{d})$$

$$f_{11}(\mathbf{d}) = 40 \cdot \mathbb{I}_{\text{ls}}(\mathbf{d})$$

$$f_{12}(\mathbf{d}) = 50 \cdot \mathbb{I}_{\text{yahtzee}}(\mathbf{d})$$

$$f_{13}(\mathbf{d}) = \mathbf{1}^\top \cdot \mathbf{d}$$

$$\mathbf{f}(\mathbf{d}) = \big(f_1(\mathbf{d}), f_2(\mathbf{d}), \ldots, f_{13}(\mathbf{d})\big)$$