UNO: A COMPREHENSIVE ANALYSIS OF REINFORCEMENT LEARNING BASED AGENT

Gaurav Sharma
Dept. of Computer Science and Engg.
Indian Institute of Technology, Ropar
Rupnagar, India
2021aim1008@iitrpr.ac.in

Ishan Tripathi
Dept. of Computer Science and Engg.
Indian Institute of Technology, Ropar
Rupnagar, India
2021aim1009@iitrpr.ac.in

I. Motivation

Over the past few decades, there has been something quite common and rather foundational in the field of Artificial Intelligence (AI). It started with Checkers and quickly gained popularity with chess and is now a multifaceted branch of AI – Game Playing. It can be imagined as giving toys to an infant to make them learn about the real world and gain problem-solving abilities.

II. Introduction

In this paper, we propose to build and evaluate an intelligent game playing agent which learns to master the game of Uno. The card game Uno is relatively easy to understand yet an overwhelming environment of 10^{163} feasible combinations makes it hard to explore for any given decision-making agent.

A discrete state-action matrix is used to train the Reinforcement Learning based agent, and the techniques used are Q-Learning and Monte Carlo. The rational agent is expected to perform well and win against its opponent despite the several sources of uncertainty involved throughout the course of the game.

III. Literature Review

Reinforcement Learning refers to the learning technique which makes the agent intelligent by exploiting its past experiences. The decision-making agent is expected to interact with the environment by taking some feasible action with the in-built goal of maximizing the cumulative sum of rewards.

The work done in this paper is inspired by [1] in which the author trains a Reinforcement Learning based decision-making agent by making it compete against a random agent for several iterations. Also, the author implements Q-Learning as well as Monte Carlo to achieve the above goal.

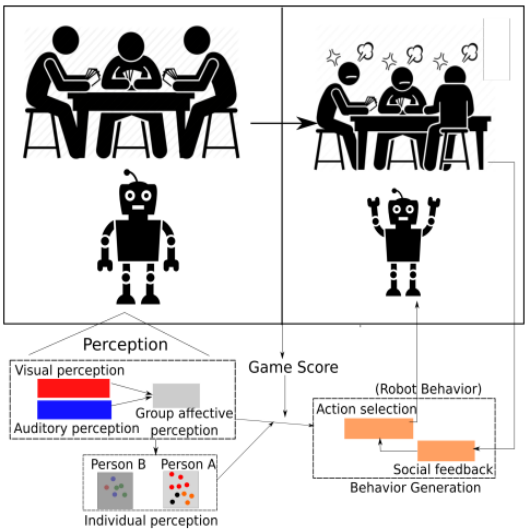

Figure 1: Role of iCub robot as a game playing agent

In a similar paper [2], the authors propose a roadmap for the development of a Human-Robot Interaction (HRI) scenario. The *iCub* Humanoid robot is used to perceive the players' audio/visual cues besides their facial expressions and body movement. Based on these observations, the *iCub* Humanoid robot

is expected to suggest an action in a competitive game scenario.

In the paper [3], we see the application of Reinforcement Learning on the game of Spades using a constrained feature map. It was observed that the accuracy of the RL-based agent wasn't too impressive primarily due to the complexity of the game of Spades.

IV. Preliminaries

I) Uno

The turn-based card game consists of a deck of 108 cards consisting of four distinct coloured cards split into the following three categories:

1. Numbered cards: 76 cards numbered in the range of 0 to 9
2. Action cards: 24 cards consisting of *Skip*, *Reverse*, and *Draw 2*
3. Wild cards: It can be played in any turn irrespective of the open card on top of the playing deck

The Uno deck is shuffled at the beginning of each game and if it gets reset if it gets exhausted during a game. The goal of the game is to play all the cards in one's had as early as possible.

II) Reinforcement Learning

It is a Machine Learning (ML) based training approach which is built upon the idea of rewarding the desirable behaviour and punishing the undesirable actions.

III) Q-Learning

It is a model-free Reinforcement Learning algorithm to learn the value of state-action pairs for a specific state at a time. It evaluates the quality of each feasible action from the current state and tries to find the best action among them.

IV) Monte Carlo Technique

It is an episodic approach to perform Reinforcement Learning which seeks to generate random samples and then evaluate them to find a pattern which it then tries to learn. In this scenario, the random component is the return or reward received.

V. Proposed Model

It is crucial to replicate the complexities of Uno while building the game environment, otherwise the agent might fail to perform when unwinding the intricacies of such an overwhelmingly complex the environment.

The Uno environment is built keeping in mind the hierarchy of components which interact with each other, as shown in the given figure. These components are briefly explained below:

- **Card:** Each card has two attributes - colour and value. It can either be playable or unplayable
- **Deck:** A collection of a variety of cards having a combined size of 108
- **Player:** It follows a strategy and interacts with the environment based on the current state and the set of actions feasible from the given current state
- **Turn:** It follows the actions of the active player and its opponent and updates the environment when required
- **Gameplay:** It represents a round of Uno as an episode, i.e. series of turns.
- Simulations: It represents the multiple rounds of Uno played between a given set of players

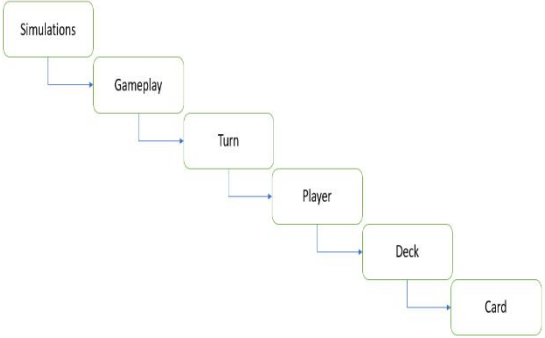

Figure 2: The hierarchy in Uno

The game environment is built around the concept of a finite Markov Decision Process (MDP), which has the following characteristics:

- **States:** Each combination in the game can be seen as a separate state
- **Actions:** It represents the behaviour of an agent to interact with the environment
- **Rewards:** An agent receives a positive or negative reward depending on the action taken at a given state.

The state definition has a length of 17 and consists of:

- *1:* Colour of the open card
- *2-5:* Total numbered cards of each colour
- *6-8:* Total action cards of each colour
- *9-10:* Total wild and wild draw 4 cards
- *11-14:* Total playable numbered cards of each colour
- *15-17:* Total playable action cards of each colour

The reward function used for Q-Learning based agent depends on previous action and is given as:

$$Q(s,a) = Q(s,a) + \text{step size} * (\text{reward} + Q(s', a') - Q(s, a))$$

The reward function used for Monte Carlo based agent is derived after the entire episode is over and is given as:

$$Q(s,a) = Q(s,a) + \text{step size} * (\text{reward} - Q(s, a))$$

VI. Setup

The model starts learning from scratch by default. The term epsilon refers to a threshold under which the agent is expected to perform randomly, i.e. the agent selects a (feasible) action at random instead of leveraging the knowledge gained from its experience. This behaviour exhibits exploration and prevents the agent to get stuck in a local optimum. The term step size denotes the learning rate used to update the Q values.

The parameters used for training using Q-Learning are:

- Total Simulations: 10,000
- Epsilon: 0.2
- Step Size: 0.2

The parameters used for training using Monte Carlo method are:

- Total Simulations: 400
- Epsilon: 0.2
- Step Size: 0.2

VII. Results

The results discussed below seek to evaluate the success of developing an intelligent agent, as well as the following results can also be used to compare the Q-Learning approach with the Monte Carlo approach of training the agents.

I) Dominance Analysis

The Q-Learning based agent had a 53.53% win rate against a random agent, whereas the Monte Carlo based agent had a win rate of 54.75%.

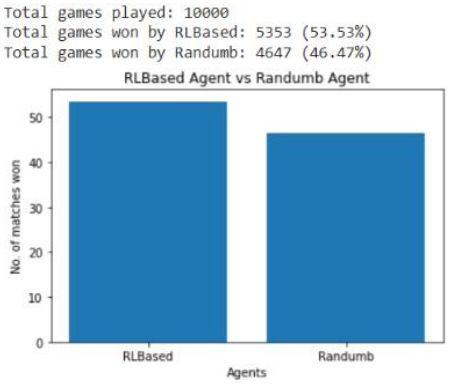

Figure 3: Dominance of Q-Learning based agent

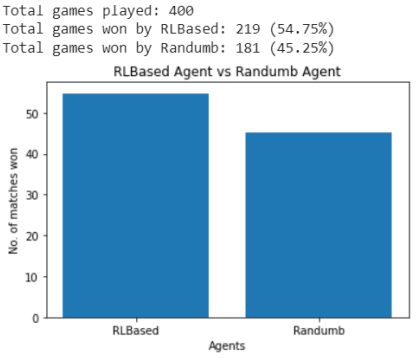

Figure 4: Dominance of Monte Carlo based agent

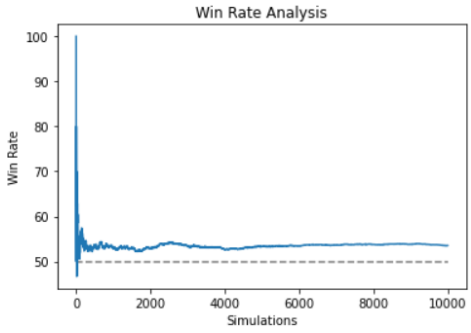

Figure 5: Win Rate analysis of Q-Learning based agents

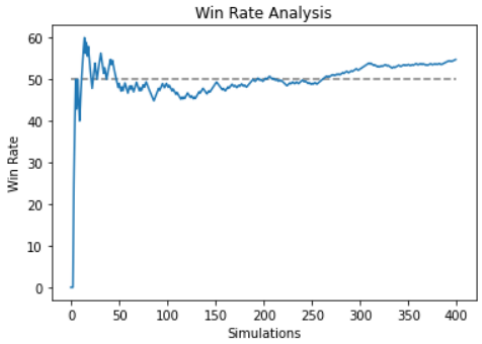

Figure 6: Win Rate analysis of Monte Carlo based agent

II) Total Turns per Game

On average, the total number of turns required by the Monte Carlo based agent in a round of Uno (~26) is approx. 9 turns less than the number of turns required by the Q Learning agent (~35).

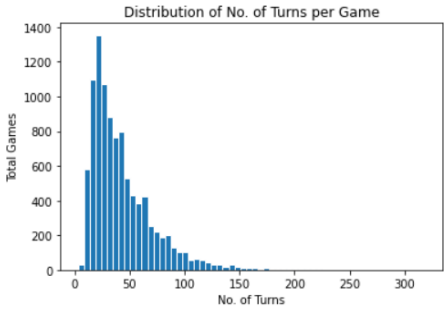

Figure 7: Turn Distribution for Q-Learning based agent

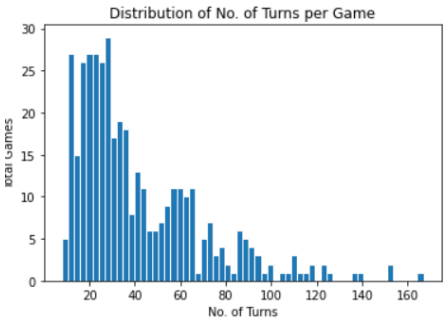

Figure 8: Turn Distribution for Monte Carlo based agent

III) Total states covered

Monte Carlo based agent resulted in much higher Q-Value coverage than the Q-Learning based agent.

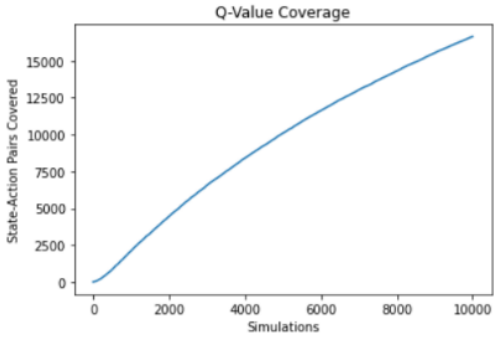

Figure 5: Q-value coverage of Q-Learning based agent

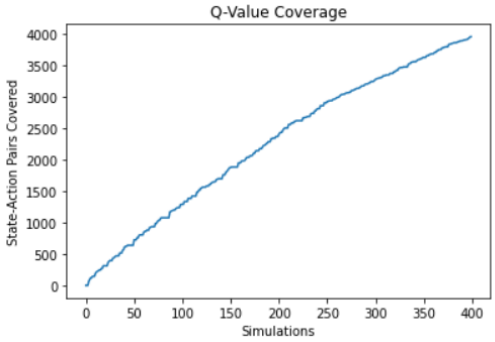

Figure 5: Q-value coverage of Monte Carlo based agent

VIII. Conclusion

After analysing the observations mentioned above, we can conclude that Uno turned out to be primarily a 'Game of Luck'. Also, a round of UNO between two players was found to last for approximately 36 turns. The player playing first was observed to have a slight advantage over its opponent, however the assumption that both the players to behave in a likewise manner plays a critical role in observing these results. Moreover, it was observed that as the agent gained better knowledge of the environment, not only was it able to dominate its opponents but also it was able to win games much quicker, i.e., in lesser number of turns. Therefore, in a game where randomness plays such a vital role, Monte Carlo approach can be declared to be a slightly better approach. However, it takes too long to train the agent as compared to the Q-Learning approach.

IX. Future Work

The above results have paved the way for future along the lines of:

- Considering the number of cards in the opponent's hand when forming a state of the Uno environment
- Implement the training using a decay on the value of epsilon as we seek to lower the random behaviour in a static environment considering the agent becomes more knowledgeable with experience
- Extending the above setup of Uno to more than two players/agents

X. References

1. '*Tackling the UNO Card Game with Reinforcement Learning*' by Bernard Pfann published in 2021.
2. '*Towards Learning How to Properly Play UNO with the iCub Robot*' by Pablo Barros, Stefan Wermter, Alessandra Sciutti published in 2019.
3. '*Reinforcement Learning on Spades*' by T Culhane.