Playing UNO with Reinforcement Learning

Surya Narayan

AE4350

SID: 6122981

31 August 2024

Introduction

An interesting use of reinforcement learning (RL) would be developing a controller that plays a card game. This shows how an agent explores the possible moves and conforms to the rules of the game. In this case, an RL controller was developed to play the game of UNO. UNO is a popular turn-based game whose objective is to empty one's hand of cards by discarding cards according to certain rules. This paper aims to explain the process of developing this controller. The first section would briefly introduce the rules of UNO that were followed and any modifications that were made to the rules. That would be followed by a section which explains the setup of the environment, agent, and reward system. Next is a section describing the results and any analysis done on the results. The last section would cover the conclusion and future steps.

Rules and Simplifications

The basic premise of UNO, as mentioned in the introduction, is to empty the set of cards in one's hand. This set of cards is called a "hand" in this report. Every player starts with 5 cards from a deck of 108 cards which have a color and a value that represent the card. The color could be red, blue, green, yellow, or wild, and the value could be any number from 0 through 9. The value of a card could also be one of the "special" values being skip, reverse, draw two, and draw four.

The deck is shuffled, and the top card of the pile is the "opening card" and is placed in the "discard" pile. The first players can discard a card from their hand which is either the same color or value as the card in the discard pile. This is followed by the next player and the discard pile keeps growing. The game is played in a clockwise direction and in the case a player does not have a valid card to play, they must draw a card from the deck and their turn is skipped.

Special cards do the following:

- Skip: Skips the turn of the player after the current player
- Reverse: Reverses the direction of play
- Draw two: Forces the player after the current player to draw 2 cards
- Draw four: Forces the player after the current player to draw 4 cards

The cards whose color is labeled as "Wild" can be played on top of any card on the discard pile and give the player the ability to change the color of play to whatever they want it to be.

A simplification made in this version is that if the opening card is a special card, then it is put in the back the deck and a new opening card is chosen. Also, draw two and draw four cards cannot be stacked on top of each other, if played the next player should strictly draw the corresponding number of cards. Finally screaming "UNO" is not required when one's hand only has one card left.

Set Up

RL Components

The agent in this case is the player that would be controlling their own play. The actions this agent can take is placing a card and drawing a card. The action taken can be represented by 3 numbers at max:

1. Discard or draw a card
2. In case it discards, pick which card to discard
3. In case the discarded card is a wild, choose a color

The state would be the agent's hand of cards, number of cards in the agent's hand, number of cards in everyone's hand, and the card on top of the discard pile. The agent makes its decision based on the mentioned state conditions. If there's a q value for a given state, it would take that action, otherwise it would explore and play a randomly chosen playable card.

Rewards for this controller are set up in the following way:

1. Negative reward for drawing a card
2. Positive reward for discarding a card
    a. Higher the reward for discarding higher the number of cards in agent's hand
    b. Added reward if the agent blocks the next player if they are close to emptying their deck
3. Massive reward for emptying their hand

The agent follows a basic Q learning algorithm and updates the q-value table at the end of every turn. The bellman equation is used in this case.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma a' \max Q(s',a') - Q(s,a))$$

The environment is the playing field with the deck, discard pile, player hands, direction of play etc. The class that describes the environment is explained in the program components section after this.

Program Components

The class that defines the environment is "UnoGame" and consists of the functions that create the deck, shuffles the deck, deals (or draw for the player) cards, checks if a card can be played, does the necessary actions that special cards bring about, and most importantly conducts the gameplay. This environment can be played both by automatic controllers such as an agent or a bot but can also be played interactively by a human. A hybrid of both is also possible in case a person wants to play with a bot or a trained agent.

Then there's a "Player" class which simply shows the hand of the player and plays cards from said hand. This class is also a parent to two other classes which are the "Agent" class and the "Bot" class.

The agent class plays as mentioned above in the RL components. However, the bot class simply plays the first playable card it has in its deck.

Other than those classes, there is the "GameResults" class which collects the game results which is the winner, the actions taken, and the final state of the game when finished.

Progress, Results, and Analysis

Initially, the agent was given an option to either choose to play or draw a card freely, because sometimes in a real game, a player strategically draws a card. However, during exploration, the agent was drawing too many cards. To try and overcome these two methods were used namely, making the reward for drawing even more negative and then making sure the agent can only play if its hand size is under a certain threshold. However, even after those modifications, the agent stuck very close to the given threshold and kept drawing an undesirable amount of times. Finally, the option to do a "strategic draw" was taken away from the agent and the final version can only play a card if it has playable cards.

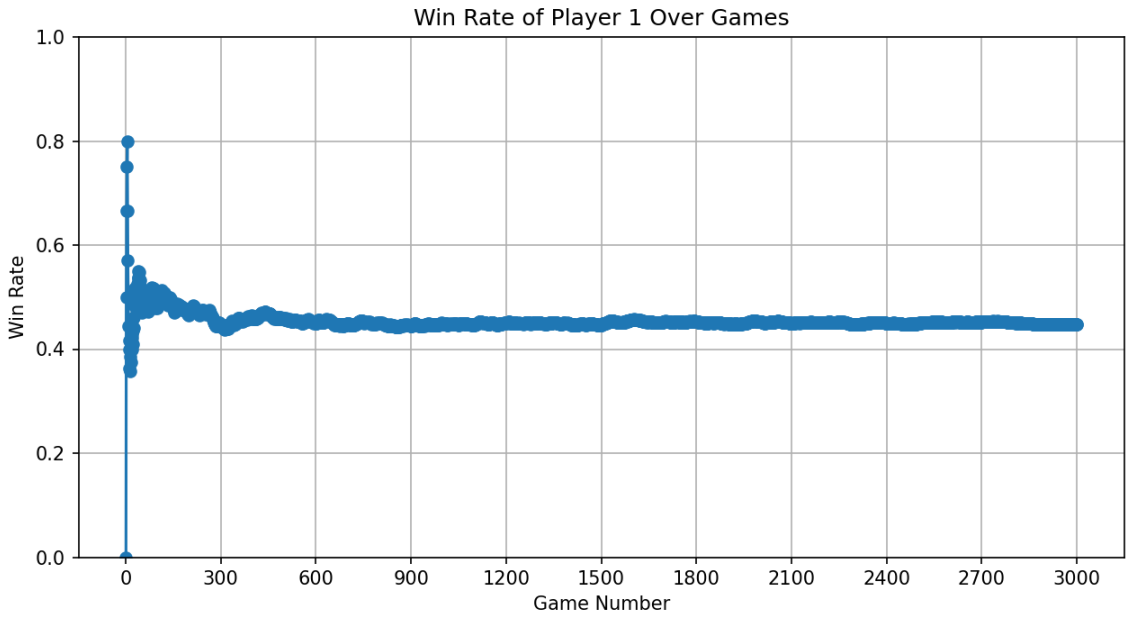

Figure 1

Figure 1 represents the win rate of the agent against the hardcoded bot over 3000 games and the win rate tapers off at 50%. Meaning that overall, the agent is a bit worse than the hardcoded bot.

Improvements and Conclusion

This RL Controller for UNO is not yet fully learned. The controller can be improved if the reward system is fine tuned a bit more. Also, there are attributes for the q learning system for the bellman equation that can be tweaked for better results. Finally, there needs to be more episodes of training.

In conclusion, the agent is a player in the UNO game environment and takes actions based on the current state. Which is just the agent's hand, number of cards in agent's hand, number of cards in everyone's hands, and the card on top of the discard pile. This agent was pit against a basic bot for a short while and seems to have almost caught up to the basic bot with a win rate close to 50%.