Deep Learning Project - Final Report
Deep Reinforcement Learning On Blackjack

Netanel Hugi 203553490
Shimon Hagag 311367536

January 6, 2019

1 Abstract

In this project we built a Deep RL model that will teach the computer (our agent) how to play blackjack. The project goal is to reduce player's loss, by finding the best strategy for a blackjack player, which will increase the odds of the player to win the dealer. Despite our learning technique, the odds are still in favor of the casino, so our goal is only to reduce the player's loss.

2 Introduction

Blackjack(1) is a card game (also known as Twenty-One). It is a comparing card game between usually several players and a dealer, where each player in turn competes against the dealer, but players do not play against each other. In our game environment only one player plays against the dealer. For the project we used the Blackjack game environment that was provided by OpenAI Gym platform(2).

2.1 Blackjack rules

In this game we need to obtain cards with sum as near as possible to 21 without going over. Each card has value, number cards count as their natural value, the jack, queen, and king (also known as "face cards") count as 10 and aces are valued as either 1 or 11 according to the player's choice. The cards are dealt randomly (with replacement). The games with 2 cards for the player and the dealer (but at first, we can see only one card of the dealer). The player can take more cards (hit) until he decides to stop (stand) or he exceed 21 (bust). After the player decides to stand, the dealer reveals his second card and takes more cards until he reaches 17 or more. If the player was bust then the dealer win, if the dealer bust the player win. If both were not bust, then those whose higher sum wins (if their sum is equal it is a draw). The reward for the results is: +1 for Win, 0 for draw, -1 for loss, +1.5 for blackjack (ace and 10/face card).

3 Project description

The OpenAI Gym platform gives us the dataset we need in terms of observations and actions. The observation space is in size of (32,11,2):

- 32 - Possible values of sum of the agent's cards (we will only use the range of [4-21]).
- 11 - Possible value of the visible card of the dealer (we will only use [A,2-10] as possible values)
- 2 - Player has usable card [0, 1].

In addition to the observation space, we have an actions space in size 2: hit or stand. The small number of possible states and actions allows us to use Q-Learning technique to train the agent. In addition, we will use exploration and exploitation so that the agent will learn better.

3.1 Q-Learning

Q-learning(3) is a value-based Reinforcement Learning technique. The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances. In this method we will use a dictionary(as a Q-table) that will include all the possible states in the game as a key, and for each state 2 possible actions as a value. Each state in the dictionary is a tuple of 3 variables (P,D,A), Where P represents the sum of the player, D is the sum of the dealer, and A represents whether the player holds an Ace. To learn each value of this Q-table, we'll use the following Q-learning algorithm:

1. Initialize Q-values ($Q(s, a)$) arbitrarily for all state-action pairs.
2. For life or until learning is stopped...
3. Choose an action (a) in the current world state (s) based on current Q-value estimates ($Q(s, \cdot)$).
4. Take the action (a) and observe the the outcome state (s') and reward (r).
5. Update $Q(s, a) := Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$

At the end of the learning process the agent can use the table to determine the optimal action for him.

- **Description of the formula:**

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

  - **Learning rate(α):** Determines how much the agent learns.
    From the formula we can see that if an α is close to 0 then it relies more on the old value, whereas if α is close to 1 then it relies more on the current information it has learned.

– **Discount factor(γ):** Determines the importance of future rewards. A factor of 0 will make the agent "myopic" (or short-sighted) by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward.

3.2 Exploration vs. Exploitation

At the beginning of the learning process we would like to explore more of the actions that the agent can perform, and as we proceed we will want to Exploit the knowledge we have accumulated. In order to implement this idea, we define a variable(ε), which will start high (1.0) and the agent will learn, the variable will be reduced to a certain value (0.1). At each step of the agent, it will choose a random number between 0 and 1, if the number obtained is greater than ε, the agent will select a random action to explore. else, he will select the optimal action according to the Q-table.

4 Experiments and results

Before the learning process we built two automatic game agents, this is to use them to simulate different game strategies, and to see the statistics of some player. We ran the experiments several times for different collections of random values, and the results we received are the average of the different values.

- **Random agent -** The purpose of this agent is to simulate the situation of a player who does not know how to play blackjack at all. In each step that he takes he randomly chooses whether to hit or stand. We ran the agent for about 10,000 games, and we got that the average loss of the agent in every 1000 games is about 350 points.
- **Auto agent** - The purpose of this agent is to simulate a "not random" player. When this agent is reaches sum of cards greater than 17 he stands. With this agent, we saw an improvement in the results. His strategy reduced the player's loss to an average of 280 points per 1000 games.

Q-Learning agent

This agent is the goal of the entire project. The agent uses the techniques mentioned in the previous section to learn how to play blackjack and create a good game strategy that will help him reduce the losses of the previous agents. The agent played for 5000 games to study the game, after the training we reached a better average result of a loss of only 100 points per 1000 games.

5 Conclusions

In conclusion, as we saw in the previous section, our Q-Learning agent managed to "get out of the casino" with a small loss (compared to the other players). In fact, the learning of our agent has created for us a basic game strategy, which can be described as the table that appears in the next sub-section. It is important to emphasize that blackjack is a lucky game, and despite our training and learning, the strategy we receive will not necessarily reach the same results in every round of games.

Strategy table

The following table describes a game strategy we received for a round of games in which the agent's loss was 60 points in 1000 games.

	A	2	3	4	5	6	7	8	9	10
	Without usable ace									
4	S	**S**	**S**	**S**	**S**	S	S	S	S	S
5	H	H	H	H	H	H	H	H	H	S
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	S	H
8	H	H	H	H	H	H	H	H	H	S
9	H	H	H	H	H	H	H	H	H	H
10	H	H	H	H	H	H	H	H	H	S
11	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H
13	H	S	H	H	H	H	H	H	S	H
14	S	H	S	H	S	H	S	H	H	H
15	H	H	S	S	S	H	H	H	S	S
16	S	S	S	S	S	S	H	S	S	H
17	H	H	S	S	S	S	H	S	H	H
18	S	H	S	S	S	S	S	H	H	S
19	S	S	S	S	S	S	S	S	S	H
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S

	A	2	3	4	5	6	7	8	9	10
	With usable ace									
12	S	S	S	S	S	S	S	S	S	S
13	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	S
16	H	H	H	H	H	H	H	H	S	H
17	H	H	H	H	H	H	H	H	H	S
18	H	H	S	H	H	H	H	H	H	H
19	H	H	S	H	S	H	S	H	H	S
20	S	S	H	S	S	H	S	S	S	H
21	S	S	S	S	H	S	S	S	S	S

As you can see, the table is divided into 2 cases (if the player has an Ace in hand or not). The table rows describe the player's current sum, and the columns describe the dealer's visible card. The values of the table is: **S:** stand or **H:** hit. Also the table contains only the values that are relevant to us (i.e., if there is no player, then the smallest possible sum is 4, and if so the smallest sum is 12).

From the table you can learn that if the sum is smaller, then the player will want to take a card, and for a larger sum he will want to stand. It is easy to see that for the low values in each section (4 and 12) the agent is wrong. This is because he finds it difficult to learn these situations (probabilistically, the agent has a lower chance of encountering them during the game).

References

[1] Blackjack: Introduction and rules.
https://en.wikipedia.org/wiki/Blackjack

[2] OpenAI gym: Game environment.
https://gym.openai.com/envs/Blackjack-v0/

[3] Q-Learning: Description and terms.
https://en.wikipedia.org/wiki/Q-learning