Machine Learning Engineer Nanodegree

Capstone Proposal

Pradhyo Bijja
August 2nd, 2017

Proposal

Domain Background

This project will attempt to teach a computer (reinforcement learning agent) how to play blackjack and beat the average casino player. The results will be compared with results of well-known blackjack strategies [1]. Mathematicians have been researching Blackjack for over 60 years [2] because of its simple rules, its inherent random nature, and the abundance of “prior” information available to an observant player [3].

Recently, I visited Las Vegas for the first time and Blackjack was one of the most popular games in the Casinos. This reminded me of the movie *21* I had watched a few years ago where some students of MIT devise a strategy to make a lot of money by counting cards and playing as a team. I had just learned Reinforcement Learning and naturally, I wanted to apply it to Blackjack to find the best strategy to lose the least amount of money.

Problem Statement

The aim of this project is to produce a Blackjack strategy that will earn more than the average casino player.

Blackjack Rules for this project

Blackjack is a card game where the goal is to obtain cards that sum to as near as possible to 21 without going over. They're playing against a fixed dealer.

Here are the rules of the game:

Face cards (Jack, Queen, King) have point value 10. Aces can either count as 11 or 1, and it's called 'usable' at 11. This game is placed with an infinite deck (or with replacement). The game starts with each (player and dealer) having one face up and one face down card.

The player can request additional cards until they decide to stop or exceed 21 (bust). After the player sticks, the dealer reveals their facedown card, and draws until their sum is 17 or greater. If the dealer goes bust, the player wins. If neither player nor dealer busts, the outcome (win, lose, draw) is decided by whose sum is closer to 21.

The reward for winning is +1, drawing is 0, and losing is -1.

Datasets and Inputs

All the data needed for this project will be simulated with the help of Open AI Gym's Blackjack environment.

At the beginning of each round, the dealer and the player (our agent) are dealt two cards each - the environment simplifies this by dealing just the values for the cards (between 1 and 10) instead of actual cards. There's a usable_ace flag which indicates whether the player has an ace card that he can 'use' as 11. So, the inputs for the agent are sum of the agent's cards, the face up card of the dealer and whether or not the agent has a usable ace. After the agent makes a decision, the environment responds with the reward for that set of inputs and decision.

Solution Statement

Decaying epsilon-greedy Q-learning will be used to solve this problem as the number of states is reasonably small.
- Possible values of sum of agent's cards [2, 21] = 20 - Face up card of dealer [1, 10] = 10 - Player has usable card [0, 1] = 2

Size of the state space is 400.

A Q-table is built for all state-action pairs and after taking an action at the end of each round, its corresponding entry in the Q-table is updated based on the reward received. The learning process stops when epsilon decays to a value less than or equal to a tolerance value. At this point, we would have the optimized Q-table which is the strategy the agent has learned to play blackjack.

Benchmark Model

Assuming the average player uses no strategy and makes a random choice each time (most likely while drunk), the payout at the end of 1000 rounds is simulated in the Open AI environment. This would be the benchmark against which the trained agent in this project will be compared.

Evaluation Metrics

Upon learning the optimized Q-table, a simulation similar to above will be carried out using the optimized Q-table. The payout at the end of 1000 rounds will be calculated and compared with the payout at the end of 1000 rounds for the average player. The agent's payout should be bigger than that of the average player at the end of 1000 rounds.

Project Design

The first step in this project would be to run the Open AI gym's Blackjack environment for the average player - taking a random action each round. After running this for 1000 rounds, the payout is calculated - this will serve as the benchmark after implementation of Q-learning.

Next, decaying epsilon-greedy Q-learning will be implemented and the optimized Q-table found.

Finally, Open AI gym's Blackjack environment is run again but this time the decisions are made using the optimized Q-table. The payout at the end of 1000 rounds is determined and compared with the simulated payout for the average casino player.

References

[1] The Evolution of Blackjack Strategies
[2] The Optimum Strategy in Blackjack
[3] A Markov Chain Analysis Of Blackjack Strategy