Blackjack AI - Presentation 1

Genetic Algorithm Progress Report

Presentation Contents

Overview

This project consists of training an agent to play blackjack as effectively as possible via reinforcement learning. How well the agent can play blackjack will be measured by the net gains/losses made by the agent over a set amount of games. The money/betting aspect of blackjack operates as a built-in reward function, which is part of the reason why this topic was chosen for the project.

The agent will learn to play blackjack via a genetic algorithm. This means we will devise a chromosome that can function as an expansive decision tree of sorts that will cover all possible states within any given game of blackjack. The agent will then read the state of the game, then use the chromosome as a kind of lookup table in order to decide which action should be taken.

Thus the primary objective is to find the optimal chromosome that can net the highest gains over a set amount of games. We will obtain this chromosome via our genetic algorithm, which will make chromosomes compete against each other based on the reward function over and over again until we arrive at a unanimous decision.

If this all sounds complicated right now don’t worry, some of the key terms will be expanded upon in the next few slides to help give you a better idea of the process that will be taking place.

What is a Genetic Algorithm?

A genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions (the chromosome), in order to find the optimal population of solutions.

In the genetic algorithm implementation we'll be using for this project, we will start with two randomly populated chromosomes, use them to play a set amount of games of blackjack, and then select the "winner" which will be whichever chromosome performed better according to the reward function. This reward function is of course just how much money has been won/lost by the agent while making decisions based on the chromosome.

Then the winning chromosome will compete against a new chromosome. This process will then be repeated thousands of times on thousands of randomized chromosomes. The best chromosomes will stay alive and continue competing against each other, eventually converging on the optimal solution.

What exactly is a Chromosome?

A chromosome (also sometimes called a genotype) is a set of parameters which defines a proposed solution of the problem that the genetic algorithm is trying to solve.

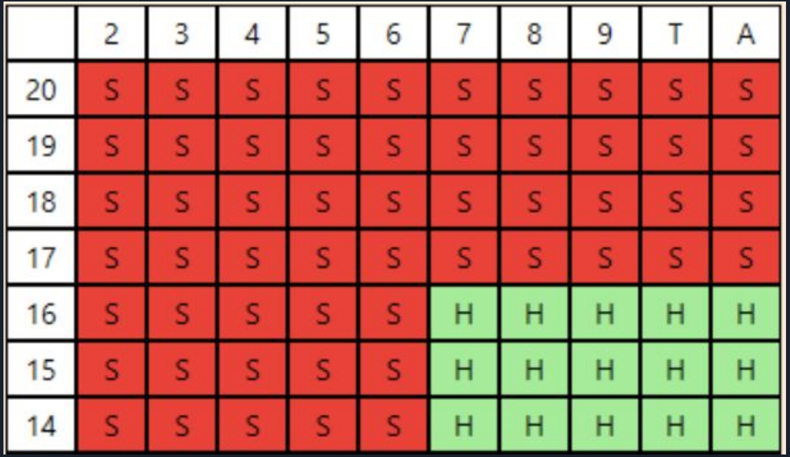
A chromosome within the scope of this project is essentially just an extremely expansive decision tree / lookup table that covers all possibles states within a game of a blackjack.

We have narrowed down the states to a 10 x 34 table. This is based on the possibility of the ten different possible dealer upcards (Jack, Queen, and King all count as ten in blackjack so they are all treated as the same card), and 34 possible hands that the agent can possess. Each of the individual indices in the 10 x 34 table will be populated with one of four possible decisions (Hit, Stand, Double Down, or Split).

Each of the 340 spots in the table, once populated with a decision, is called a “cell.” The chromosome is the table containing all 340 cells. Our goal is to obtain the chromosome with optimal cell structure.

Chromosome Basic Example

Below we can see a small cutout of a portion of a potential chromosome for the project. The x-axis (2 through A) represents the possible dealer upcards. The y-axis represents possible player hands. Since this is a just an example I only included 7 of 34 possible hands. The red and green boxes represent individual cells of the chromosome. S represents the “stand” action, and H represents the “hit” action. As you can see this particular chromosome recommends standing with any hand value over 16 regardless of the dealer upcard, but once player values get lower the recommended actions begin to depend more on the dealer hand more.



	2	3	4	5	6	7	8	9	T	A
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H

Greater Detail

Information on the projects chromosome and possible actions can be found in greater detail in the first section of my project’s readme.md file within the github repository at:

www.github.com/tenjotenge/blackjack-ai

Genetic Algorithm Process

Randomize

Randomly populate a chromosome with possible actions.

Play

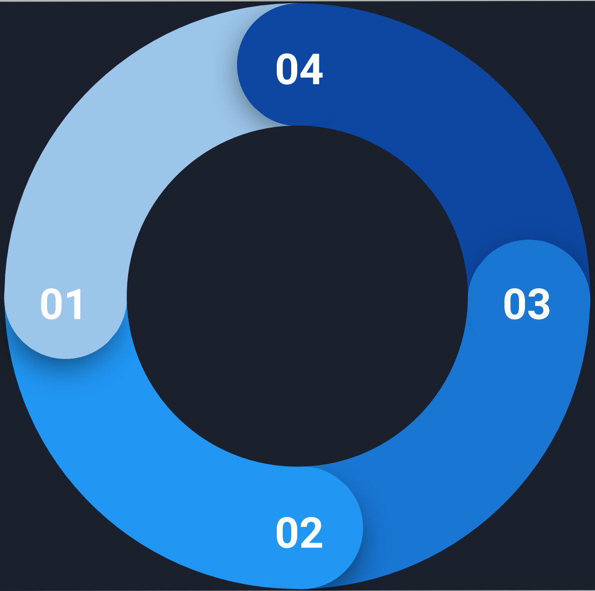
Let the agent play a set amount of games with that chromosome.

Advance

Select the chromosome that performed better and repeat the process.

Compare

Compare gains/losses against another chromosome.



Primary Objectives

01 Create a functional and appropriate chromosome structure that the agent can use, as well as an environment for the agent to use it in.

02 Create the genetic algorithm to refine this chromosome via the “tournament” process of making the chromosomes compete against one another.

03 Use the genetic algorithm process and its version of reinforcement learning to derive the optimal chromosome.

Secondary Objectives: Why?

Before listing the secondary objectives we thought it best to explain the rationale behind these implementations.

Blackjack, like all forms of gambling that take place in a casino, is a game in which the house maintains a slight advantage over the player. This means that even with the optimal 10 x 34 chromosome, it is impossible to win money over a large enough amount of games. No matter how much you optimize the optimize the 10 x 34 chromosome, you will always lose money on a large enough sample size.

So this means that the only way to train an agent to play blackjack effectively enough for it to be worth your while and to come out with positive net gains, you will have to incorporate the three features listed in the next slide in order to widen the agent’s advantage.

Secondary Objectives (if we have time after deriving the optimal 10 x 34 chromosome)

01 Expand the chromosome to incorporate more states than the initial 340.

02 Add weights to the chromosome based on counting cards and the current likelihood of the next card to come off of the deck.

03 Add functionality to range bets based on the current card count and the agents perceived odds of winning the next game.

Software/Hardware Requirements

The software requirements for this project at the moment consist solely of Python, but I may need to incorporate a framework like TensorFlow/PyTorch in order to implement the features listed in Secondary Objectives.

As far as hardware requirements go, I am currently just running everything off of a Macbook and haven't had any issues. I could expedite the process with more compute but I don't see it being necessary at the moment. If I end up requiring more compute and additional hardware for implementing the secondary features I will give updates on that in the final presentation.

Project timeline

9/15	10/1	10/15	11/1	11/15	12/1
Topic Selection	**Structuring**	**First Presentation**	**Making Adjustments**	**Wrapping Up**	**Final Presentation**
Narrowed down potential projects to a handful of card and board games, last two options were poker and blackjack.	Chose a method of training the agent and began setting up an initial structure for the project and genetic algorithm. Begin training the agent.	Preparing and giving the first presentation while also monitoring the training process and making adjustments as needed.	The initial training process should be just about finished up by this point and from here we can make adjustments and expand features and functionality as needed.	The vast majority of the project should be finished up by this point, and there will only be time for small modifications and beginning work on the final presentation and deliverables.	Presenting the finished product in its entirety as well as turning in all documentation and the codebase.

Project Details and Codebase

Information on the project as well as the current Python code I have so far can be found in greater detail within the github repository at:

www.github.com/tenjotenge/blackjack-ai