Introduction

Artificial Intelligence, also known as AI, is a powerful contemporary tool that is starting to integrate into various fields, including image classification, speech recognition, gameplay, and more. To be specific, AI has been applied to games such as Poker to be able to beat the best human player. To expand on this, Professor Luo aims to integrate artificial intelligence with decision-making algorithms. The objective is to develop a computational strategy that consistently outperform human players over a series of games.

Research & Learning Process

Given my familiarity with Java, I initially developed a simple Java-based rock-paper-scissors game between a computer and a player. The process of implementing this game started with the computer randomizing an integer ranging from 0 to 2, with 0 representing rock, 1 representing paper, and 2 representing scissors. Correspondingly, the player would choose an integer also ranging from 0 to 2.

Then, to calculate the success rates, I iterated this basic algorithm to execute 200 times. I then tested it against various player strategies, including random moves, cyclic patterns (e.g., inputting 0, 1, 2 in sequence), and the repeated use of the same move sequence.

Algorithms & Results

1. Follow The Leader

This algorithm's strategy stores the number each time the player plays either rock, paper, or scissors. Depending on the most frequently played move, the algorithm selects to play the counter move. E.g., rock: 10 instances | paper: 5 instances | scissors: 9 instances. Based on the frequency of moves, the algorithm selects paper to counter the most frequent move, rock.

2. Hedge

The Hedge Algorithm predicts the players' moves by calculating the probability rock, paper, or scissors will be played. Initially, there are three experts, each representing the probability of each move. After each round, the experts are updated according to the loss suffered from predicting the player's move and the move that was played. To calculate the success rate, the Hedge Algorithm has to be manually tuned with η, representing the learning rate.

3. Ada-Normal Hedge

Ada-Normal Hedge Algorithm is similar to the Hedge Algorithm, but it excludes η, the learning rate. Another difference is the computations when calculating the loss each expert suffers. Using the Ada-Normal Hedge Algorithm, the loss suffered from predicting the player's move and the move that was played is defined.

4. Hedge-Weighted Tree

This algorithm pieces together mini subtrees to create a tree data structure. It stores the player's move in an array list and updates it after every round. Starting a new round, it begins at the root of the tree and traverses through each node following the context of the array list. The leaf nodes, located at the bottom of the tree, hold the experts of each move and uses hedge to predict and compute the probability of the player's next moves.

5. Sleeping Experts

This algorithm is also similar to the Hedge-Weight Tree algorithm. However, instead of taking the weights of the leaf nodes and using Hedge to calculate the probability, it takes the collective weights of all the nodes that were traversed through. Building a tree with this algorithm is also different than the Hedge-Weight Tree. Instead of building subtrees and piecing it together, this algorithm creates new nodes in real-time as it traverses the tree.

Results Analysis

- All the algorithms have a difficult time predicting the player's next moves when playing randomly.
- All the decision-making algorithms' performance in predicting the player's moves is substantially better than the basic Math.random() algorithm.
- The results show that when players have a strategy in playing their moves, these decision-based algorithms are able to easily pick them up. This results in the computer winning a majority of the plays, which was the goal of this project.