Test Report
Rock-Paper-Scissors Algorithm
26.11.2024

Background:
The algorithm I chose to test out (algorithmv3.py) is a five-level Markov Chain. It uses five separate dictionaries (could be combined into one, but accessing each turned out to be more time demanding per cycle than the current implementation), and each turn each one is cycled through.

Each algorithm stores a series of past moves, and looks for any possible repeating patters (such as how many times a player chose scissors (2) after playing a rock twice (00) → 002: 3, and if a hit is found, then the algorithm records that the player has just played two rocks (00) and will now play a counter move, assuming that the player plays scissors (2) next. For this reason, the algorithm becomes more accurate the longer you play, and also works better against human players, than for example a random number generator, which often have a long enough seed, where repetition is sparce.

The reason as to why the algorithm repeats every single Markov-chain depth for each move, is that the computer also keeps track of which depth is performing the best at winning. At first, no five-depth moves can be made, so the two-depth algorithms are preferred. However, after around 50-rounds, the longer depths are far more accurate, and become far more relevant. For this reason, each one must be tracked for each move. This turns out to not be too demanding, as the dictionary method chosen is computationally light as compared to lists, or accessing a longer shared dictionary.

Tests done:
The algorithm was tested against human-players, as well as against a random number generator (see bottest.py) and the results show that the algorithm shows promise against human-opponents. It is able to beat the average human who is consciously choosing moves and using a gameplan with relative success, against three players who played five games of 50 moves each, it won an average of 3.5 times, tied 1 time, and lost 0.5 times.

The odds were different once users were told how the algorithm works, where educated players were able to get ahead of the algorithm relatively quickly.

Against a random number generator, the algorithm performs as well as wound be expected, in 25 repeats of 2,000,000 rounds, the algorithm won an average of 667,000-rounds, lost 663,000 and tied 670,000; indicating a slight difference from a simple random number generator vs random number generator of 1/3 odds of each.

Conclusions:
While the algorithm is not perfect and can easily be beat once its core function is stripped down bare, it does what I set out to do with it. There is no way (apart from cheating) to create an algorithm to compete and successfully win a human-player each time, in a game of chance. Any sort of probabilistic algorithm has to make the assumption that player actions are independent from the moves of the algorithm itself, however I feel as if the five-depth Markov chain is able to better adjust to the interplay of human and machine, just like a human-opponent in the real world.