**THEORY OF COMPUTATION AND COMPILER DESIGN PROJECT**

**on**

**Predicting Human Move in Rock, Paper and Scissors Game using Machine Learning Techniques**

By: **ArpitRathi** (16BCI0065)

**Parmeet Singh** (16BCE0184)

**INTRODUCTION**

Rock, Paper and Scissors is a hand game usually played between two players in which the two players simultaneously forms one of the three shapes with their hands. It is a zero-sum game i.e. either there is a tie or one of the player wins.

This being a study of the human mind and the pattern in which humans think to make their next move is an interesting area for using algorithms and machine learning concepts to predict the next move and beat humans in this game using an algorithm.

Because of this many algorithms have been made in the past to simulate this issue. The most popular of which is the Iocaine Powder, which won the first International RoShambo Programming Competition and uses a heuristically designed compilation of strategies.

**ABSTRACT**

This project focuses on creating a bot that can play the common game of Rock Paper Scissors (RPS) better than its human opponent. The key idea is that humans tend to play in patterns rather than completely at random. Using machine learning techniques, we will train a bot to learn how we play RPS, thus giving it an advantage over us. For example, if every time we play the sequence of ‘Rock’, ‘Rock’, ‘Paper’ and ‘Rock’ our next throw is consistently ‘Scissors’, then our bot learns that and plays ‘Rock’ in the next game to beat us.So, for this problem our input is a sequence of n human plays, which we represent as an n length vector, where each element in the vector is an integer in [1,3]. ‘Rock’ is represented by 1, ‘Paper’ is represented by 2, and ‘Scissors’ is represented by 3. Given this sequence of human plays, our bot uses various classifiers to predict our n+1 th play, which is also represented by an integer as described above.Given this play, the bot chooses its winning counterpart (i.e. predicting ‘Rock’ entails playing ‘Paper’,predicting ‘Paper’ entails playing ‘Scissors’, and predicting ‘Scissors’ entails playing ‘Rock’).

We will also create different difficulty level:

1. **Easy** – just a random guess out of the three possibilities.
2. **Medium** – based on previous inputs by the user and the maximum probabilities of the user to choose their next move.
3. **Hard** – In addition to the checking the previous inputs it will also keep a track of the current player playing style and pattern and make its next move.

**LITERATURE SURVEY**

1. **A Bayesian Model for Plan Recognition in RTS Games Applied to StarCraft**

The task of keyhole (unobtrusive) plan recognition is central to adaptive game AI. “Tech trees” or “build trees” are the core of real-time strategy (RTS) game strategic (long term) planning. This paper presents a generic and simple Bayesian model for RTS build tree prediction from noisy observations, which parameters

are learned from replays (game logs). This unsupervised machine learning approach involves minimal work for the game developers as it leverage players’ data (common in RTS). We applied it to StarCraft1 and showed that it yields high quality and robust predictions that can feed an adaptive AI.

1. **Value-function reinforcement learning in Markov games**

Markov games are a model of multiagent environments that are convenient for studying multiagent reinforcement 17 learning. This paper describes a set of reinforcement-learning algorithms based on estimating value functions and resents 18 convergence theorems for these algorithms. The main contribution of this paper is that it presents the convergence theorems 19 in a way that makes it easy to reason about the behavior of simultaneous learners in a shared environment. Ó 2001 20 Published by Elsevier Science B.V.

**STRATEGY**

The question arises as to how the previous results and previous gestures influence the players’ next gestures. We propose that there are 10 potential strategies that a player can adopt in a multi-round game of rock-paper-scissors. An example of one strategy is: a player repeats a gesture when they win, alternates gestures when they lose and chooses randomly when they draw. Thus for each of the three results, there are three potential responses (repeat, alternate, random). There are therefore nine (three x three) potential strategies. Another example of this kind is when a player alternates their gestures when they win, chooses randomly when they lose and alternates their gestures when they draw. The tenth strategy is to identify which of the nine strategies the other player has adopted and then to counter that strategy. For example, if an opponent tends to repeat their gestures when they win and if they had won the previous round with the gesture of ‘rock’, then a player would counter their strategy by using ‘paper’ in the next round.

It is difficult for a human to accurately determine which strategy their opponent is playing. In order for them to do so they would need to keep track of eight variables:

* The previous gesture
* The previous result
* A count of the number of times their opponent alternates after winning
* A count of the number of times their opponent repeats after winning
* A count of the number of times their opponent alternates after losing
* A count of the number of times their opponent repeats after losing
* A count of the number of times their opponent alternates after drawing
* A count of the number of times their opponent repeats after drawing

We will observe the computer learning by picking a strategy and sticking to it for a while. Here are a few things to try:

* Pick Rock, then Paper, then Scissors, then Rock again and keep that pattern up. See how quickly the computer learns to beat you every time?
* Having done that a few times, change strategy and pick Paper 5 times in a row. See how the computer spots your change of strategy and alters its play?
* Pick any strategy of your own and see if the computer can spot the pattern.
* See if you can be perfectly random in your choices and beat the computer.

**CURRENT BOTS**

**Iocaine Powder**:This bot was created by Dan Egnor for a RPSbot competition in which bots played against each other. Iocaine Powder uses a combination ofthree strategies to predict its opponent’s next move: guessing randomly, playing against the mostfrequently used throw, and history matching (i.e. finding patterns in the opponent’s history).However, this bot’s edge lies in the fact that it also detects the opponent’s meta strategy (i.e. guessing one ahead, two ahead, etc…).

**MegaHAL**:Thisbot was created by Jason Hutchens for the same RPS competition described above. His approachwas to create a simple Markov model that stores frequency information about the opponent's playsfor all possible contexts. This allows the bot to predict the next play in the form of a probability distribution over all possible throws. The bot then picks the throw that maximizes the expected value of its score (where a win scores a 1, a tie 0, and a loss 1).It also only tracks the frequencyinformation over a sliding window, in case the opponent changes strategies over time.

**METHODS TO BE ADAPTED**

**SYSTEM DESIGN**

Our project shall contain three parts:

1. Easy – It will randomly choose one of the three choices and produce the output. So there is 50% chance of this bot winning based on sheer luck.
2. Medium – In this mode, the bot will keep a track of the various choices made buy all the users till date and come up with its next move based on the track record kept by it.
3. Hard – In this mode, the bot in addition to the medium mode will also keep a track record of the current player playing pattern and predict his next move keeping in mind the record pattern kept by it.

**Language Chosen to be used** : Python/C++.

**REFERENCES**

1. http://ofb.net/~egnor/iocaine.html
2. http://web.archive.org/web/20020802183909/http://www.amristar.com.au/~hutch/roshambo/
3. Jordan MB. An actuarial artificial intelligence for the game rock-paper-scissors. S Afr J Sci. 2016;112(5/6), Art. #a0154, 3 pages. http://dx.doi. org/10.17159/sajs.2016/a0154