**Contact information**

Keegan Gunderson

Mat Hirschi

Colton Moore

**Introduction**

The source of data we analyzed is data that was created by players each time rock paper scissors was played. Using the past moves of players will allow us to predict the next move that should be played by the user. Overall we wanted to create/use an algorithm that could train a computer to pick up on real world tendencies of a player and use those tendencies to beat them.

**Data Preparation**

The data set we chose was creating a game of rock paper scissors and putting this data through an algorithm to see if it can consistently beat the player who is playing. This problem is interesting because it shows that an algorithm can pick up on human tendencies but will also predict moves before they happen so it can continually beat the player.

We initially found a dataset from the interact that has wins, losses, ties, moves used, and round number that we were going to clean and use for our project. The more we got into cleaning the data the more we realized that we wanted the rock, paper, scissors game to create the data as the user played. If we have the data created from users playing the game we will have it in the format we need to run our desired algorithm. The data created while playing was wins, losses, ties, and moves used.

**Mining/learning from data**

We though that a neural network would be the best algorithm to use when training a computer to choose the beat someone at rock, paper, scissors. The neural network would have been effective for choosing the best result in order to beat someone in rock, paper, scissors but we came across a better algorithm to use. The algorithm we decided to use were Markov chains, and a stochastic matrix, this proved to be better than a neural network because this algorithm is easier to implement and trains the computer as you play so each round the computer becomes harder and harder to beat.

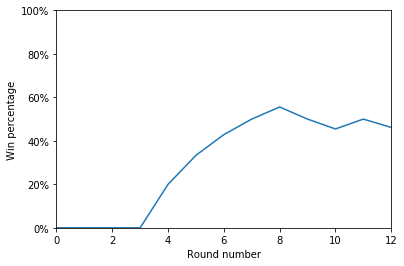
We initially started with a neural network to train the machine to beat a player at rock paper scissors. We used the machine to learn based on almost half a million data points of games from sources online. For some reason, this seemed pretty inconsistent, and would sometimes vary from 20% accuracy to 40%, but on rare occasions, up to 75%.

We decided that maybe the data was too inconsistent to have all of this data work for each individual person, we instead, we moved to Markov chains where the games learns as it goes. This seems to work pretty well, because it will learn the tendencies for each person, instead of getting general data from a huge population. It’s main goal is to beat the person it’s playing against by learning THEIR patterns and tendencies.

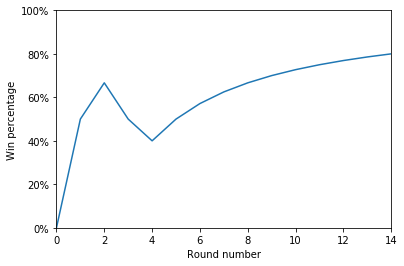
**Results**

Using Markov chains, one player cannot use an obvious pattern without the computer catching onto it. There are no TRUE random patterns, so even if a player tries to choose random patterns while playing against the computer, the computer still might find a hidden pattern that the user is oblivious to and could still possibly beat them. This seems to work well, since the graphs showing the average win percentage seems to keep growing to a little over 50%. This is not always the case, as the Markov chains isn’t a perfect learning method, as with every learning method to some degree.

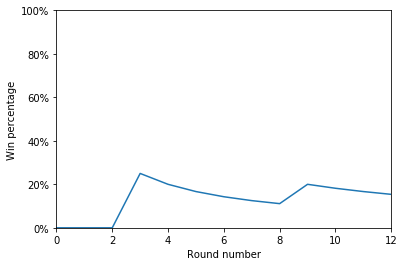
Game 1 results (Played with repeating options, then changed how I chose my options)



Game 2 results (Repeated with the options of “rock”, “paper”, and “scissors”, in that order)



Game 3 results (We tried to be truly random)



**Conclusion**

This learning algorithm performs well with obvious patterns from the user. When the player tries to do random options, sometimes the machine does well, but sometimes it doesn’t do well. It depends on if the user’s “random” patterns have some hidden pattern to them or if the pattern actually is hard to see.

Over the course of multiple games with over 20 rounds each, the average end win% turns out to be close to 60%.

**Lessons Learned**

Not all machine learning algorithms work the same or are as easy to implement.