Machine Learning in Qualitative finance CS 381
Final Project Report
Group 3
Teammates:Farrukh Hanif, Sumaita Hussain , Eric Chou , Oakga Lwin

## Introduction

The project is a replicated project based on higher order neural network analysis in EUR/USD currency exchange trading. However, since the research already lay us the group work on which neural network performance is better in this category, we decided to explore on the top neural network as described in the paper. These are Softmax cross entropy network, Gaussian mixture network and Recurrent neural network. The Higher Order neural networks such as PSI sigma is touched in these studies but however; due to the lack of resources available (research studies or libraries) we were not able to replicate such network.

## Data

The data in the paper is not available as well as ancient and thus, we decided to find similar data dating from 2000 and onwards. We have recreated a data set that will be similar to the one in the paper from multiple databases by mix and matching with respect to each date. We used currency data as variable input to our neural network and we pick our currency data that will be related to USD since this will be our primary forecasting variable: Chinese yen to USD, AUS to USD, PND to USD, CND to USD, JPN to USD etc.

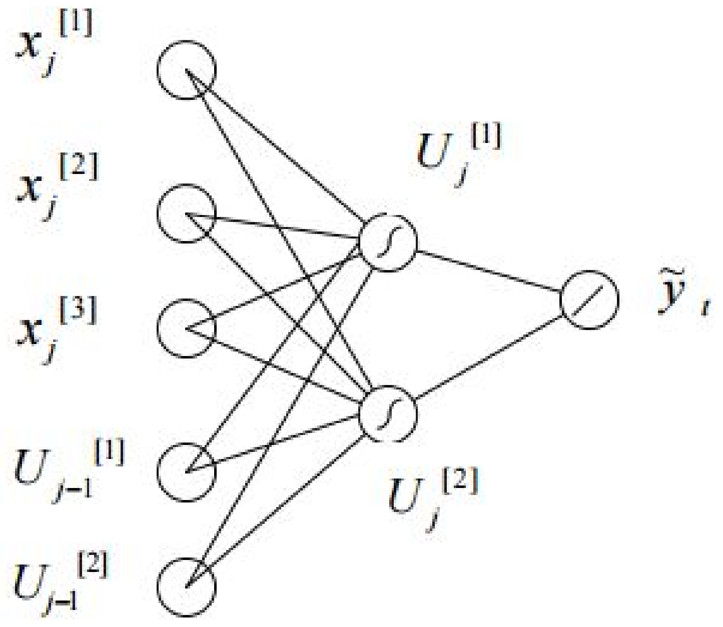
## Neural Network Specification and building Process

The tools that we used for the neural networks are mainly Weka, deeplearingj4 and Simbrains libraries. All neural networks are built with an input layer, output layer and hidden layer although the number on layer on hidden layer may varies from each network-to-network.

## Recurrent neural network

For RNN we were lucky enough to find the same weka plug-in that was used by the researchers.
The plugin we found was based on research by Elman in the 1990s.

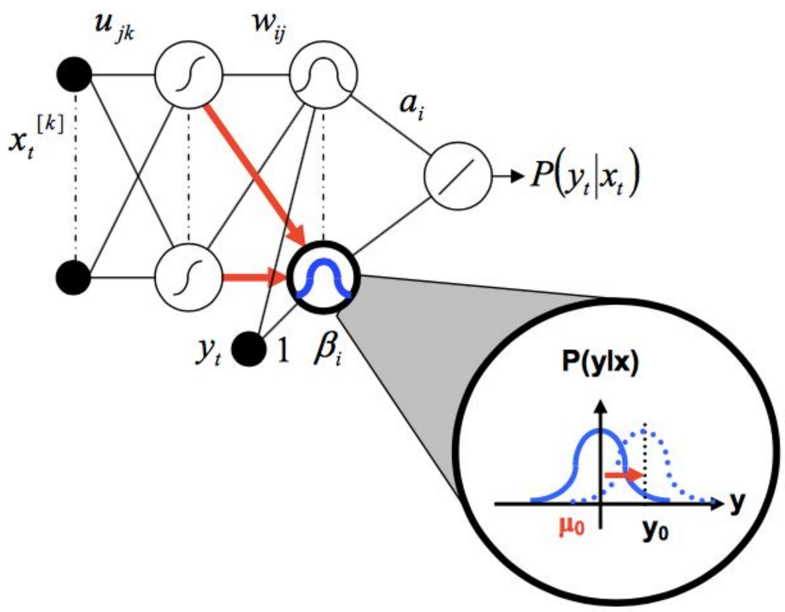
The RNN used in the paper was described as.



For our research we used the same structure of the RNN. However, we used 5 model inputs. We used 2 hidden nodes just like in the paper. The challenging aspects of gathering our data was to combine 6 different data sets into 1. Weka was not great when it came to reading inputs of different formats such as json. Thus, we were forced to convert our data into weka's format. Another issue we ran into was the fact that we were using a classifying algorithm. This algorithm would give us a output state, however it wouldn't give us a numerical output.

We got around this problem by converting our numerical data into three different states. The states we chose were positive, negative, and neutral. If EUR/USD conversion was higher than a previous day we would put positive. If it was lower, we would put negative. If the price was the same, we would put neutral. The script written for this purpose will be attached.

Gaussian Mixture neutral network

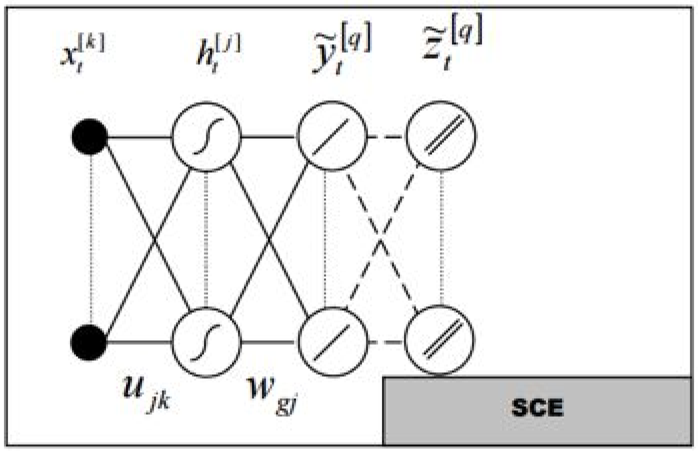


The Gaussian Mixture Model works by finding the probability distribution function of the linear combination of several normal distributions. The gaussian distribution is the hidden layer. We found a similar algorithm in weka which uses gaussian distribution but is not multivariate. Similarly to the RNN, this was a classifying algorithm and the challenge was that it worked with only one attribute at a time. Because of this, we ran the attributes through the algorithm one by one. Each of the attributes produced different results, some better than others. We then took the average to evaluate how GMM works overall.

**GMM**

| **CurrEx Rates** | **Sharpe** | **MDD** |
|---|---|---|
| austousd | .02 | -1% |
| chntousd | .03 | -3% |
| cndtousd | .03 | -3% |
| jpntousd | .06 | -3% |
| pndtousd | .02 | -2% |

**Softmax Cross Entropy Model**



The layer has two output neurons which is using SCE function activation as describe as following:
For each score x in neuron

```
    Softmax_score<- Calculate exponent ( x – max(x))  / Softmax_score.sum().
```

The two hidden layers are using linear and logistic activations functions. Note that the bias weights are initialized randomly. The learning rate varies from 0.3 to 0.01 but we achieve better results with smaller learning rate from 0.4-0.7 range. On this model, we got 57% correct prediction rate.

**Methodology**
The general process of how we tested a neural network on its trading performance will be described as following: Training Process, feeding the historical data, calculating the benchmarks.

**Training Process and Test Data Period**
We have a trading days beginning from 2001 and ending in 2008 with total data points of 1749. Test Data is however, only a year worth of data starting from January to December of 2008.

**The trading process**
Our simulation of a trading process is a simple excel sheet. Our neural network outputs a pair of log file which include a date and its prediction on forecasting price state: which mainly falls into 3 categories: positive, negative and neutral. Based on this log data, for each day, we perform a transaction in size of 1$ USD. To visualize this, for every positive output, we perform a sell signal and for every negative output, we perform a buy signal. We hold onto our asset during the neutral state.
A minimum of 4 pips is allowed for the entirety of transaction timeline. After allowing for transaction costs, we calculate the portfolio value at that particular time.

**Calculating Benchmarks**
We evaluated our data based on two measures: Sharpe Ratio and Maximum Drawdown. Our data consisted of the portfolio value within a span of 365 days. This value decreased or increased based on the buy/sell signal for our respective neural network. Once we obtained the portfolio value of each day (starting from 1000 USD), we used excel to calculate the maximum drawdown, return, mean, standard deviation, and finally the sharpe ratio.

***Drawdown*** = (PortfolioValue.(*Date-1*))/Max(PortfolioValue.*All*)
*Drawdown - the ratio of Portfolio Value of previous day in proportion to the highest Portfolio Value
***Max Drawdown*** = Min(Drawdown)

***Return*** = (PortfolioValue.*Date*)-(PortfolioValue.(*Date-1*))/(PortfolioValue.*Date*)*100
*Return - the percentage change of Portfolio value from the day before to the next day
***Mean*** = Mean Return
***StDev*** = StDev of Return
***Sharpe Ratio*** = Mean/StDev

**Conclusion and Results discussion**
Our results are not similar to the ones in the paper since we received insignificant findings. The sharpe ratios at each network is low enough for reliably trust the network on actual market. This might be due to the discrepancies we described as following:

| | Recurrent | GM | Softmax |
|---|---|---|---|
| Sharpe Ratio | 0.19 | 0.03 | 0.2 |
| Maximum Drawdown | -11% | -2.4% | -10% |

Discrepancies with the previous study:
Note that the dataset we create is made to be similar as much as possible with the previous study but however, we noticed that the dataset we were using when calculates the time series data with rate of exchange follow a normal distribution curve where as the paper mentioned their data set is carefully picked to be non-normal as other studies suggested.

We allowed 4 pips for each transaction to follow the study but however these transaction costs were for the time period when the study was conducted and might not be consistent with the current time frame we are conducting our study on.

The study perform transaction of large amount whereas we perform each transaction with a limitation of 1$ and therefore, we don’t see much significant fluctuations in our profits which also means the trading was not effective.