Financial Time Series Prediction

CS726: AML Project

by

Khushhall & Kanhaiya
Raj & Aishwary

Indian Institute of Technology, Bombay

Contents

Chapter 1

Introduction

Time series arises naturally in the financial world. Usually, we want to predict the price of a stock using historical price of the stock.

Stock prices gets influenced by a very large amount of factors viz. government policy announcement, company performance, etc. Hence, the stock prices are too noisy in nature.

In this project, we use different neural network architecture to predict stock price. The method is applied to several time series and the resulting generalisation performance of the trained network predictors is analysed.

We have tried the following neural network architectures:

- LSTM
- ConvNet
- ConvNet + LSTM

We used two different sets of data.The description of the the data used is as follows:

- Scraped the data using Yahoo finance API.
- Data from Kaggle Battlefin competition.
- Data from site Ouana solutions for Business

We will now discuss the challenges that financial time series data can pose. Firstly, the data is very arbitrary. A lot of noise is generally present in this data and prediction accuracy is generally very low. This is one of those areas where

machine learning has not been able to perform well.
Since the data is non stationary, prediction from past values is a poor idea and hence there is always a large amount of error because of no periodic pattern. We try to predict data based no the fact that at certain times, the data could follow certain pattern and our aim is to minimize the error during that time.

The problems faced are obviously challenging. We have tried to work on small things as well to try and improve the performance. Some techniques used:

- To improve the results of forecasting, it is better to remove the correlations between the inputs and make them statistically independent

- Different learning algorithms are used to train the model e.g. backpropagation, Resilient Propagation(RPROP) and a version of it called the RPROP+.Using the resilient Propagation results in an increase in convergence speed by about 25 %.

Chapter 2

Data description

Kaggle BattleFin Data Challenge

We have to predict the percentage change in a financial instrument at a time 2 hours in the future. The data represents features of various financial securities (100 in total) recorded at 5-minute intervals throughout a trading day. We have been given data of total 200 trading days. Within each trading day, we have been provided the outputs(features of 100 length) as a relative percentage compared to the previous day's closing price. In total we have 55 time-stamps in the interval from 9:30am to 1:55pm and we want to predict the output at 4:00pm.

Yahoo Finance

Scraped the data using Yahoo finance API. The data consists of multi-dimensional time series related to a particular stock. Ex: One of the time series contains the closing price of a stock for a time period of 5 years. The features are exchange rates, closing price, etc. Our aim here is to predict the closing price of Yahoo stock (or any other stock).

Oanda Solutions for Business

The webpage https://www.oanda.com/solutions-for-business/historical-rates/main.html contains free accurate data for download about typical financial instruments. The data for exchange rates has been downloaded from this website.

Chapter 3

Methods Used

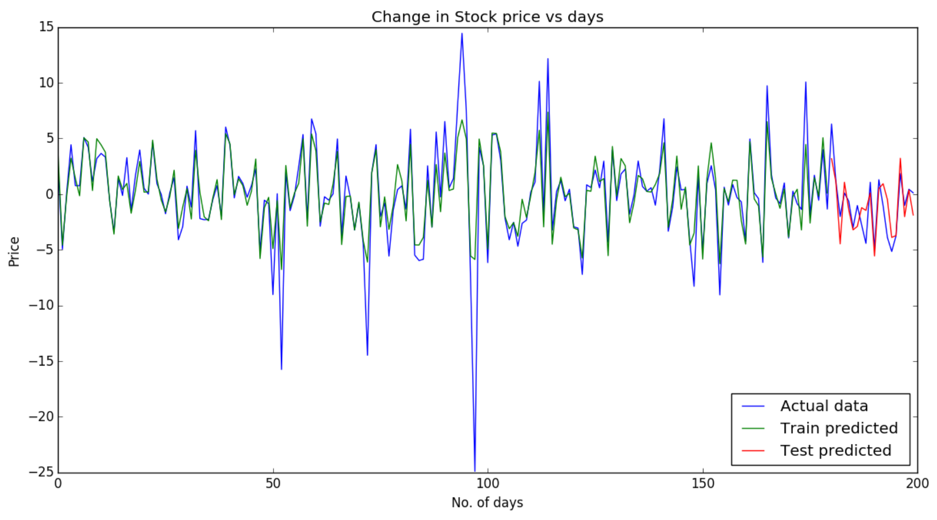
We describe here what are the different neural net architectures we have used and the corresponding results. We have used keras library to implement the neural nets

3.1 LSTM on BattleFin data

Input: 55 time stamps of feature vectors(size - 55x100)
Output: a single feature vector(size - 100x1)

Code description:

getdata()
Returns a 200x55x100 input matrix and 200x100x1 output matrix
Split the data into training(180 days) and test(20 days)
Network model
Layer1 - Lstm (with default output activation='tanh')
Layer2 - The output layer (no activation means linear)
We tried using dropout and another lstm layer but it doesn't help to improve the result
Loss function
We used mean squared error since the output entries belongs to real numbers.
Results
Training error(RMSE) as: 2.38305222603
Test error(RMSE) as: 1.06684005913
Visualizing Results Showing the time series graph of price change for a financial instrument:



We see that the predicted price change is almost similar to the actual price change.

We also tried the above code for predicting only first 20 features (instead of 100).In this case, we got test rmse = 0.83. It is lesser because now, we are predicting only 20 features for the same no.of training examples.

3.2 ConvNet on BattleFin data

Here, we think 55 time series vectors as a single image and fed it to the Convolution network architecture

Input: 55x100 image (which represents 55 time stamps of feature vectors

Output: a single feature vector(size - 100x1)

Code description:

Reading data

Returns a 200x55x100 input matrix and 200x100x1 output matrix

Split the data into training(180 days) and test(20 days)

Network model

Layer1 - 2D Convolution with output activation='relu'

Layer2 - 2D Convolution with output activation='relu'

Layer3 - 2D Maxpooling following by Dropout

Layer4 - Flattening the output

Layer5 - Feed-forward with relu activation followed by Dropout

Layer6 - The output layer (no activation means linear)

Loss function

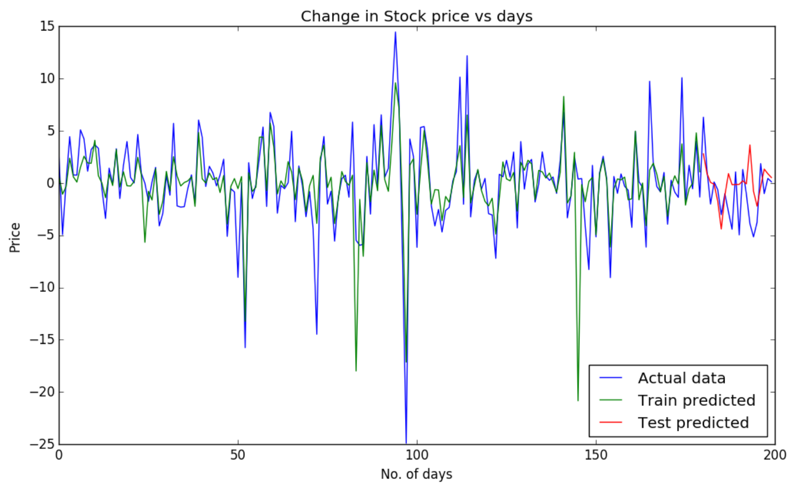
We used mean squared error since the output entries belongs to real numbers.

Results

Training error(RMSE) as: 2.46726249595

Test error(RMSE) as: 1.0668103916

Visualizing Results Showing the time series graph of price change for a financial instrument:



Wee see that both LSTM and CNN gives same performance.

3.3 LSTM for predicting exchange rate values

Here LSTM has been used to analyse and study the performance of recurrent neural network on Exchange Rates. Exchange Rates form a financial time series prediction problem where the ratio of two currencies varies over time. This data is generally very noisy and has a lot of small fluctuations. Because of this, it is extremely hard to predict future values by learning from the past data.

Input: 365×1 data (which represents 365 exchange rate values (1 year)). The data here has been taken from the website for Oanda Solutions which maintains records of historical financial data and makes them available for download.

Output: a 365 - l length vector where l is the size of the input vector to the lstm

Code description: There are 3 models. The 1st script makes an attempt at predicting data using a single layer of lstm. The second method does the same using two lstm layers. The third method uses a single lstm layer but the input vector is now the difference between the successive values.

The first method tries to see if the data can be modelled by a single LSTM and to keep the model simple. A big problem with financial time series prediction models is overfitting and hence more complicated models can overfit easily. Hence this simple model has been used.
The code has been distributed in the ratio of 7:3 for training and test.

Network model

Layer1 - LSTM layer with 'tanh' activation

Layer2 - dense (fully connected) layer with 'tanh' activation

Attempts have been made using dropout and without that (comes as layer 2).

Loss function

Mean squared error since the output entries belongs to real numbers. Although, a good loss measure would be the comparing if the predicted output and the actual output both increased or decreased on a particular day. Since a cost function for that was not available, the third model has been created.

Results

Apart from the usual RMSE, a binary accuracy has been defined and used. If the predicted value is greater than the predicted output of the previous financial day, then the output = 1 else 0. Similarly the ouput if calculated for the dataset. These binary values are now compared to get the accuracy. Why this is done? We try to see how the model works every day. Even if the model does not predict a value close to the actual data, but if it is able to predict whether the exchange rate would go up on a certain date or not, that would a very valuable information to have. This is the motivation behind defining such a accuracy measure.
The tables below show the binary accuracies and rmse errors that have been achieved for 3 different exchange rates for single and multiple LSTM models. (Note- the binary accuracies are not errors, they represent accuracy). Length of RNN tells the size of the input vector to the LSTM or the window used over the dataset.

Currency - USD - Euro
Single LSTM

Length of RNN	Binary Accuracy	RMSE Error
40	0.51	0.7
20	0.5	0.71
10	0.54	0.68
5	0.59	0.64
1	0.59	0.64

Model 2 - Multiple LSTMs

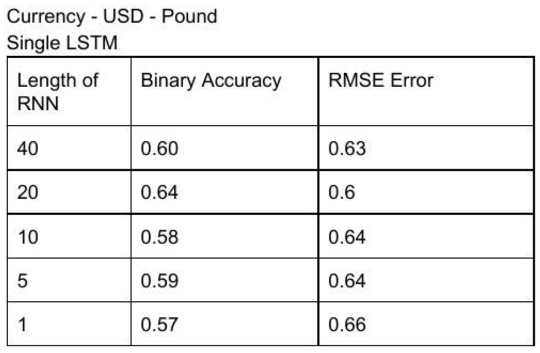
Length of RNN	Binary Accuracy	RMSE Error
40	0.49	0.71
20	0.5	0.71
10	0.45	0.74
5	0.5	0.7
1	0.55	0.67

Currency - USD - Canadian Dollar
Single LSTM

Length of RNN	Binary Accuracy	RMSE Error
40	0.53	0.69
20	0.61	0.62
10	0.63	0.61
5	0.6	0.63
1	0.62	0.61

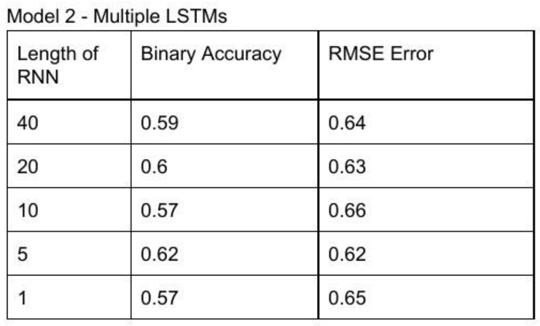
Model 2 - Multiple LSTMs

Length of RNN	Binary Accuracy	RMSE Error
40	0.6	0.63
20	0.58	0.65
10	0.6	0.63
5	0.56	0.66
1	0.55	0.67



Currency - USD - Pound
Single LSTM

Length of RNN	Binary Accuracy	RMSE Error
40	0.60	0.63
20	0.64	0.6
10	0.58	0.64
5	0.59	0.64
1	0.57	0.66

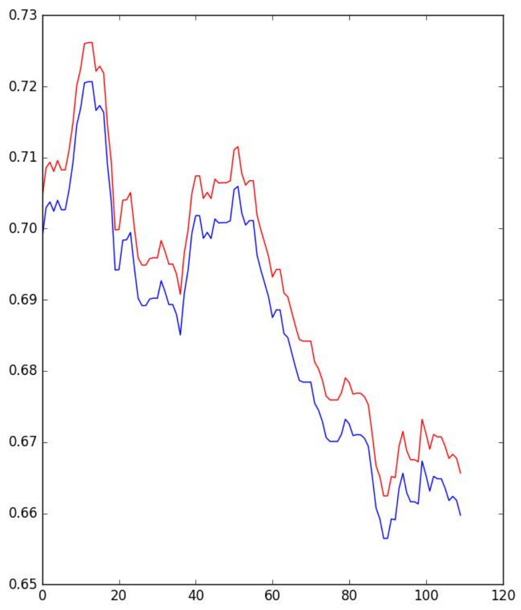


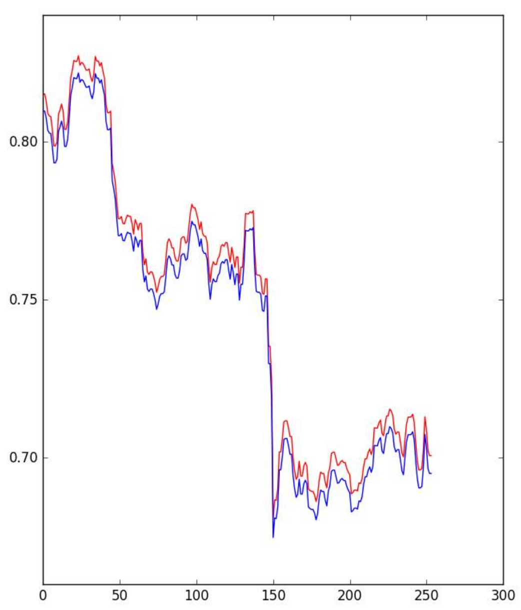
Model 2 - Multiple LSTMs

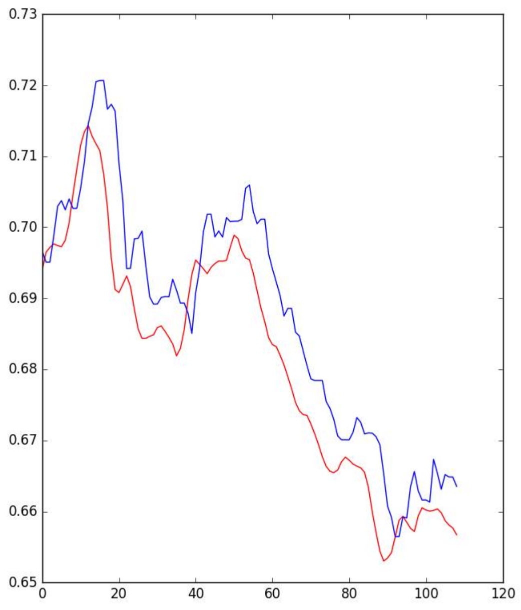
Length of RNN	Binary Accuracy	RMSE Error
40	0.59	0.64
20	0.6	0.63
10	0.57	0.66
5	0.62	0.62
1	0.57	0.65

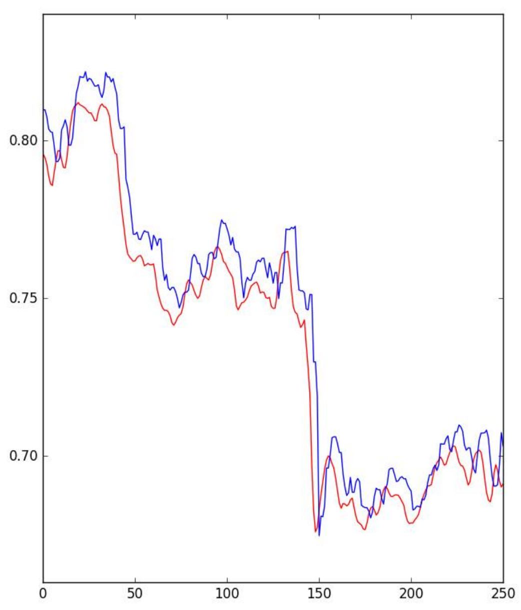
Following are the images of the

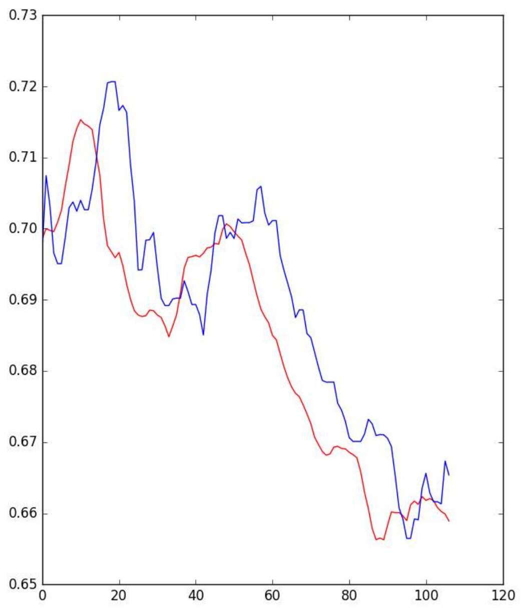
From the above data, we can see that the performance is very poor. This is in general the case with financial time series prediction and with exchange rates because of lack of a lot of data (feature vector has just 1 dimenstion).
It is observable that the accuracy for the single LSTM model decreases as we increase the RNN length. It is probably because too much past data for a simple model makes its performance poor on the recent data while a complicated model like a two LSTM model can handle it.
The two LSTM model works okay for higher sizes of RNN. The performance is found to be better for single LSTM model here even though there is not much difference.
Below are the images of the predicted and the actual values (left-test and right-training). The order of the images is for RNN length=1,5,10,20,40

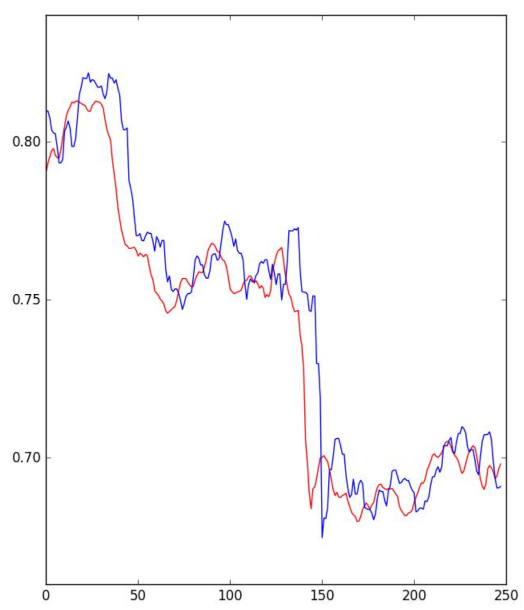

0.73
0.72
0.71
0.70
0.69
0.68
0.67
0.66
0.65
0
20
40
60
80
100
120

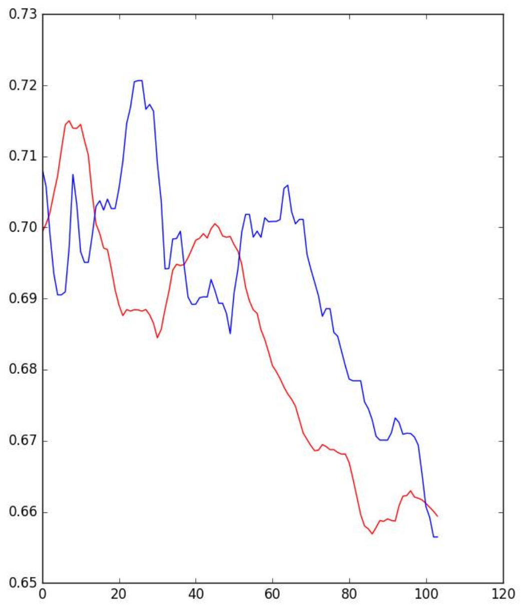

0.80
0.75
0.70
0
50
100
150
200
250
300

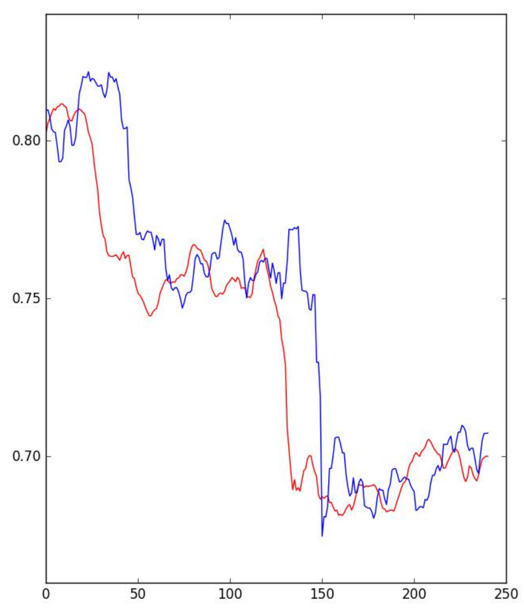

0.73
0.72
0.71
0.70
0.69
0.68
0.67
0.66
0.65
0
20
40
60
80
100
120

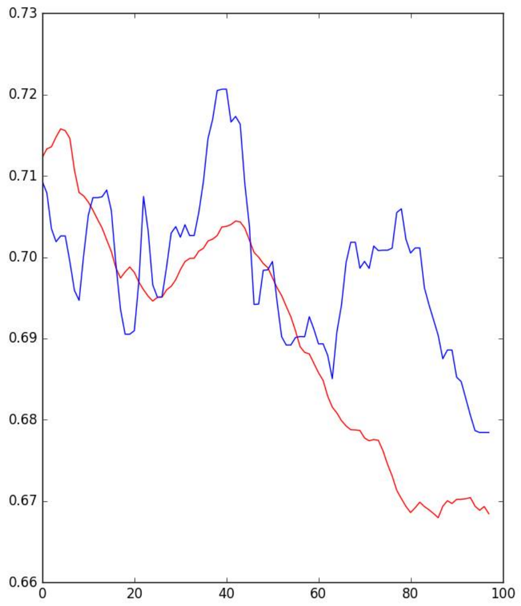

0.80
0.75
0.70
0
50
100
150
200
250

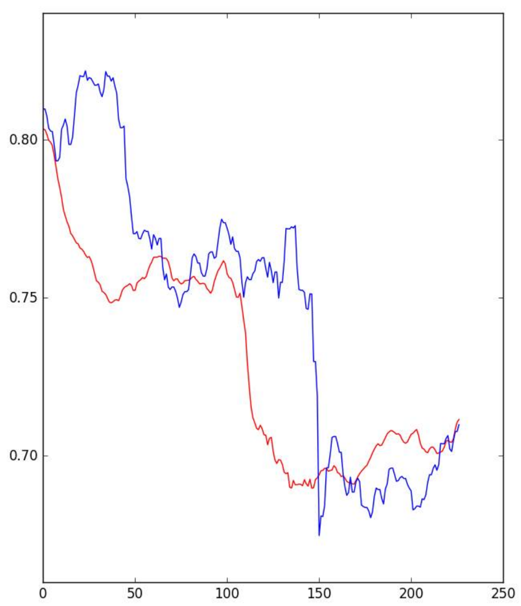

0.73
0.72
0.71
0.70
0.69
0.68
0.67
0.66
0.65
0
20
40
60
80
100
120


0.80
0.75
0.70
0
50
100
150
200
250


0.73
0.72
0.71
0.70
0.69
0.68
0.67
0.66
0.65
0
20
40
60
80
100
120


0.80
0.75
0.70
0
50
100
150
200
250

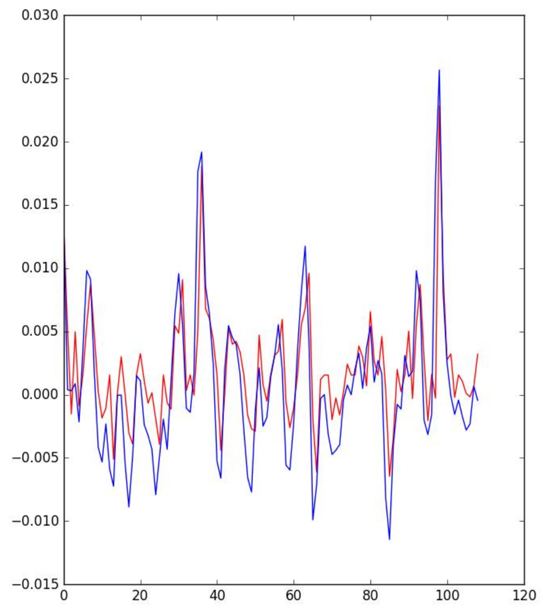




Clearly, we can see the performanc decreases as RNN length increases (1st image - RNN length = 1, 2nd image - 5, 5th image - 40).

Finally we discuss the third method. The binary accuracy is defined as follow: If the predicted price goes up (or down) and the same happens with the actual price (in the same direction, either up or down) then the error = 0 else, error = 1. The mean over the test set is then taken. The improvement in the third model is quite a lot. We take the case of only one length RNN. In this, we get a binary accuracy of 0.7 and rmse error of 0.55.

Following is the image of the actual and the predicted difference of two consecutive values. Left one is the test set while the right one is the training set.



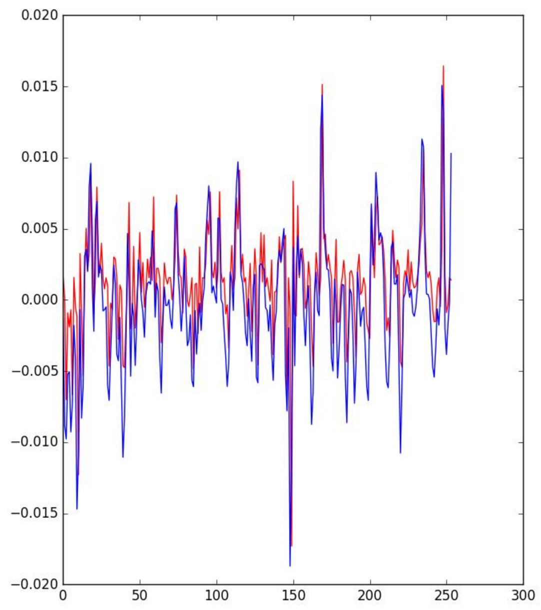


Figure above: The difference values in the predicted and the actual data.

Chapter 4

Other Approches used

Normalization
We tried normalizing battleFin data, but it doesn't help much. Since the range of our data is not too large and also the mean of every feature across time is concentrated around zero.
Softening the labels
We tried this in LSTM code for battleFin data. Instead of giving hard lebals for the training data, we tried to softened(smooth) labels. For this we apply an averaging filter across time using different window lengths. But it didn't improve the performance.
CNN and LSTM together
We tried this in cnn code for battleFin data.LSTM network was applied after Convolution network to capture the sequential information. But the result didn't improved. **Binary classification** Instead of predicting the actual price change or the exchange rates, the binary output were predicted which shows whether it will rise(1) or fall(0). But in that case the accuracy value were coming out to be around 52 % which is not an impressive result for binary classification.
Elman Neural Network Tried to use the Elman model to predict the time series. Could not successfully implement it.

Chapter 5

Resources Used

- We have used **keras** library to implement Neural network. We have used keras documentation to start from.
- All the code we have used in this project can be found **here**
- The code can also be found on Github **here**
- We used our personal laptop for running the experiments. On average our code took $\sim$ two minutes to run.
- Most of the time was spent on tinkering with LSTM models. We tried various different methods viz. using LSTM + ConvNet to get better results.
- Initially we all did literature survey based on different data sets. We then decided to work in two team. Kanhaiya & Khushhall, worked upon Kaggle BattleFin Challenge. Raj & Aishwarya worked upon exchange rate prediction. Many times there were bugs in our implementation. We helped each other to fix the bugs.