Analysis of Financial Time Series Data

Arpit Jain
14134

Arham Chopra
14130

Aashutosh Tiwari
14005

Vibhav Sarraf
14792

Mayank Singour
14378

The code can be found here at:
`https://bitbucket.org/ArhamChopra/stockprediction`

Abstract

A stock market, equity market or share market is the aggregation of buyers and sellers of stocks. It depends on large number of variables and also fluctuates a lot. The movement of prices of any stocks nearly follows a random walk distribution if observed over a small time period, however certain pattern can be observed over large time intervals. In this project we have tried to model this financial time series to predict the change in stock market values (i.e. positive or negative) on daily basis. We have made a combined model using SVM and TDNN, and then applied ensemble methods (boosting) to improve accuracy. Finally we compared the accuracy of our combined model with existing popular models used in this field, based on SVM and Neural Network. Most of the current models give an accuracy of around 60%, so we aim at increasing the accuracy through the combined model. We are predicting just the labels and not the actual values of increment or decrement because stock traders involved in intra-day(short-term) trading are mostly concerned with knowing whether their investment will be profitable or not, rather than knowing the actual values.

1 Problem statement

We will be implementing our code to predict the change in price of the NASDAQ Composite Index (stock market of USA) based on it's previous day's prices and other economic variables which are known to have significant impact on Stock Market. Essentially this is a classification problem: +1 represents increase and similarly -1 represents decrease in the price of stock from its previous day's value.

2 Related Work

Many research has been done on whether the stock prices are predictable or not. According to *Efficient Market Hypothesis* theory, stock prices only respond to new information and hence follow a random walk. If this was true than the direction of future stock prices could not be predicted with more than 50% accuracy. But there are certain models that are able to predict price direction with more than 50% accuracy. It means that the machine learning techniques are able to predict stock prices.

Most research for time series forecasting has focused on Artificial Neural Networks(ANN). ANNs contain interconnected nodes that are organized into different layers based on their function (input layer, hidden layer, output layer). The weights are assigned to these connections between nodes and as the algorithm is trained, it notices trends in the data and reassigns the weights.. ANNs are able to give quite accurate results when the data does not have high sudden variations. But due to high

variations in stock prices with time even the advanced algorithms like ANNs tend to fail. ANNs are able to give accuracy slightly greater than 50%.

Another model that is used for this purpose is (Kernalized) Support Vector Machines(SVM) in addition or as an alternate to ANNs. SVMs are one of the best known binary classifiers. If provided with carefully chosen features, SVM can give fair accuracy in case of stock market prediction. They can also be modified to account for the time-series nature of the data.

3 Algorithm

Here we describe the algorithm we have implemented in this project.

SVM

SVM creates a decision boundary that separates one type of data from the other. SVM makes sure that most of the points that belong to one category are on one side of the boundary and those which belong to other category are on the other side of boundary. A linear boundary(hyperplane) is defined as:

$$w_0 + w_1x_1 + ... + w_nx_n = w_0 + \sum_{i=1}^{n} w_ix_i = 0$$

where **x** = $x_1, ..., x_n$ is a feature vector. If for a particular point the summation is less than zero than that point belongs to one category otherwise if the summation is negative than the point belongs to other category. Hence for our purpose the labels for examples are given by:

$$y = sign(w^Tx + w_0)$$

The optimum hyperplane is the one that maximizes the distance of plane to any point and is known as maximum margin hyperplane. This hyperplane best separates the data. However the data may not be perfectly different therefore some error variables $\xi_1, ..\xi_n$ called slacks are added.

Every training example (x_n, y_n) with a slack "ξ_n" should satisfy the margin condition:

$$y_n(w^T * x_n + b) > 1 - \xi_n$$

The primary objective can be written as:

$$min_{w,b,\xi_n} f(w, b, \xi) = \frac{||w||}{2} + C\sum_{n=1}^{N} \xi_n$$

subject to constraints

$$y_n(w^Tx_n + b) > 1 - \xi_n$$

Solving the loss function, we get our final equations as:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

where α equals

$$\alpha = max_{\alpha \leq C} \quad L_D(\alpha) = \sum^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (x_m^T x_n)$$

s.t. $\sum^N \alpha_n y_n = 0$

Here C is selected in such a way so that it penalizes the model more on errors made on recent data than on past data.

$$C = (C1, C2...Cn) \;\; where \;\; C_i = \frac{c * i}{N}$$

where c is a hyper-parameter
SVMs are not limited to just linear boundaries. They can be extended to non-linear boundaries using Kernel functions. The kernel function that we used in our model is *radial kernel*. Radial kernel is the most popular kernel function and is defined as:

$$K(x, x') = exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

We also used *Adaboost* algorithm to increase the accuracy of our model.

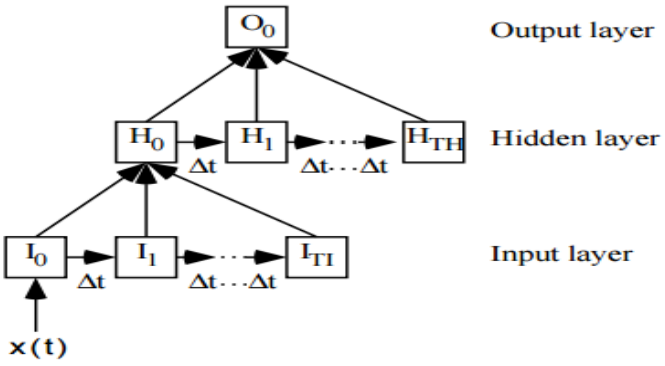
Time Delay Neural Networks

Time delay neural networks (TDNN) attempts to find a pattern in the input data sequence. After the TDNN has trained it takes the stock price of past few days and predicts the price of stock for the next day. The idea behind TDNN is to make use of the sequential information which other neural networks ignore. The TDNN units recognise features independent of time-shift (i.e. sequence position) and usually form part of a larger pattern recognition system. An input signal is augmented with delayed copies of other inputs so that the neural network becomes time-shift invariant since it has no internal state.

$$y(t) = f(x(t-1), x(t-2), ...x(t-d))$$

TDNN like other neural networks is composed of multiple interconnected neural networks. TDNNs are implemented as a feedforward neural network instead of a recurrent neural network due to their sequential nature.

A set of delays are introduced to the input to achieve time-shift invariance so that the data is represented at different points in time.



We have used Bayesian Regularization Back-propagation to train our TDNN model. Bayesian regularization automatically sets the optimal performance function to achieve the best generalization based on Bayesian inference techniques. The Bayesian optimization of the regularization parameters requires the computation of the Hessian matrix at the minimum point.

The cost function used in Bayesian Regularization is given by:

$$C(i) = \beta * E_d + \alpha * E_w$$

where E_d is the sum of squared errors, and E_w is the sum of squared weights.α and β are Bayesian hyperparameters that are used to tell in which direction (minimal error or minimal weight) the learning process must seek.

4 Feature Selection

Feature selection is an important task for SVM classification model. Here we define two terms volatility and momentum for stock price. Volatility is defined as an average of percent change in a given stock's price for the past n days.

$$Volatility = \frac{\sum_{i=t-n+1}^{t} \frac{P_i - P_{i-1}}{P_{i-1}}}{n}$$

where C_i is the closing price of stock on i_{th} day. Momentum is defined as the average of stock's price movement over past n days.

$$Momentum = \frac{\sum_{i=t-n+1}^{t} y}{n}$$

These two quantities can act as features for the given stock for our SVM model. However we increase our features by noting that the given stock may depend on other macroeconomic variables and these variables can be used as features. For each of these variables their volatility and momentum can be calculated and can be used as features in the SVM model.

For the TDNN the past day value of stock prices act as the features.

5 Code Description

We made two different codes in Matlab, one for the implementation of S.V.M and the other for implementing Neural Networks. A brief description of their design is given below. We implemented them in a way so that the output of Neural Network's algorithm can be used as a part of the input for testing our S.V.M. model, which is how is how we achieve a combined model.

Time Delay Neural Network

We used the neural network library provided by matlab for building our neural network model. We used the nnet library in Matlab, for designing various types of Neural Networks such as the feed forward and recurrent neural network.
There is a start file which runs the complete program and outputs the net which can then be used to predict the results for new data points.
In total we have built three functions: TimeSeriesNN, Test, DirectionPerformance.

1. TimeSeriesNN: This function was generated with the nnet library. We have partially edited this function as per our needs. We have added more arguments like delay, LayerArray among others to allow easy building of network with different parameters without having to run the nnet gui program. This function builds a neural network and trains it according to the input data and returns the net along with the performance of the network on train data.
2. Test : This function runs the network on input test data and outputs accuracy of the network in predicting the correct direction of price shift using the DirectionPerformance function.
3. DirectionPerformance : Used by the Test function to get the accuracy test the predicted direction shifts with actual direction shifts.

Currently we are training the neural net using *trainbr*(Bayesian regularization backpropagation algorithm). The Stopping condition is completion of one iteration over the data set or the stablization of the performance on the data and error.

Individually modelled SVM

Before making the combined model for SVM we made an individual model. So the set of features didn't include the current day's values, as without the data from NN we cannot get the current day's values for test data.So our model predicted only on the basis of previous day's values. Instead of shortlisting features like we did above, we took all the features plus an additional set of features constructed by taking difference of the current features. Let z_n be set of features of macroeconomic variables for day n and x_n be the data set constructed. Then

$$x_n = [z_{n-1}, z_{n-1} - z_{n-2}]$$

Combined Model

We developed our own code for implementing SVM rather than using one of the inbuilt SVM solver in Matlab. The inbuilt function was much faster than our own code but we faced some major problems while implementing it.

- We couldn't change the values of different hyper parameters involved in solving the S.V.M.'s which we needed to do in order to learn best possible results.
- We needed to improvise the S.V.M. code in such a way so that it could incorporate the time series nature of data.
- While running the inbuilt code we saw that the train accuracy(around 84%) was much higher than the test accuracy(around 57%), so their was no advantage in implementing ensemble methods like Boosting which works by increasing the train accuracy.

We needed to solve the dual objective problem

$$\alpha = max_{\alpha \leq C} L_D(\alpha) = \sum^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (x_m^T x_n)$$

$$s.t. \sum^{N} \alpha_n y_n = 0$$

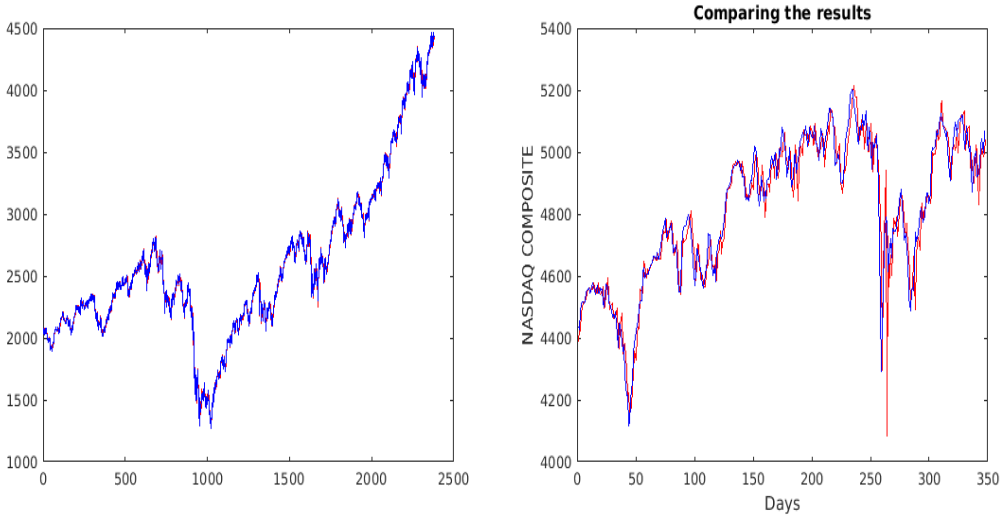
For this we used the "fmincon" function in Matlab which is used to minimize an equation subject to some constraints.

Our data set initially consisted of daily values of 16 macroeconomic variables along with the daily prices of NASDAQ composite index. We then shortlisted 4 of these 16 features which had highest correlation with the Stock Market prices. We then scaled the data so that all columns' values lie between 1-100. As x_n we took the current day and previous days' values of these features and Stock price data.These features were again transformed to produce more features so as to find the relation between different data points. But during testing the current day's prices won't be available to us(otherwise the model would become trivial), so these features will be generated from NN model.This is how our model became a combination of both NN and SVM. Values predicted by the NN model served as some of the features of SVM, which helped in achieving a better accuracy.

6 Results

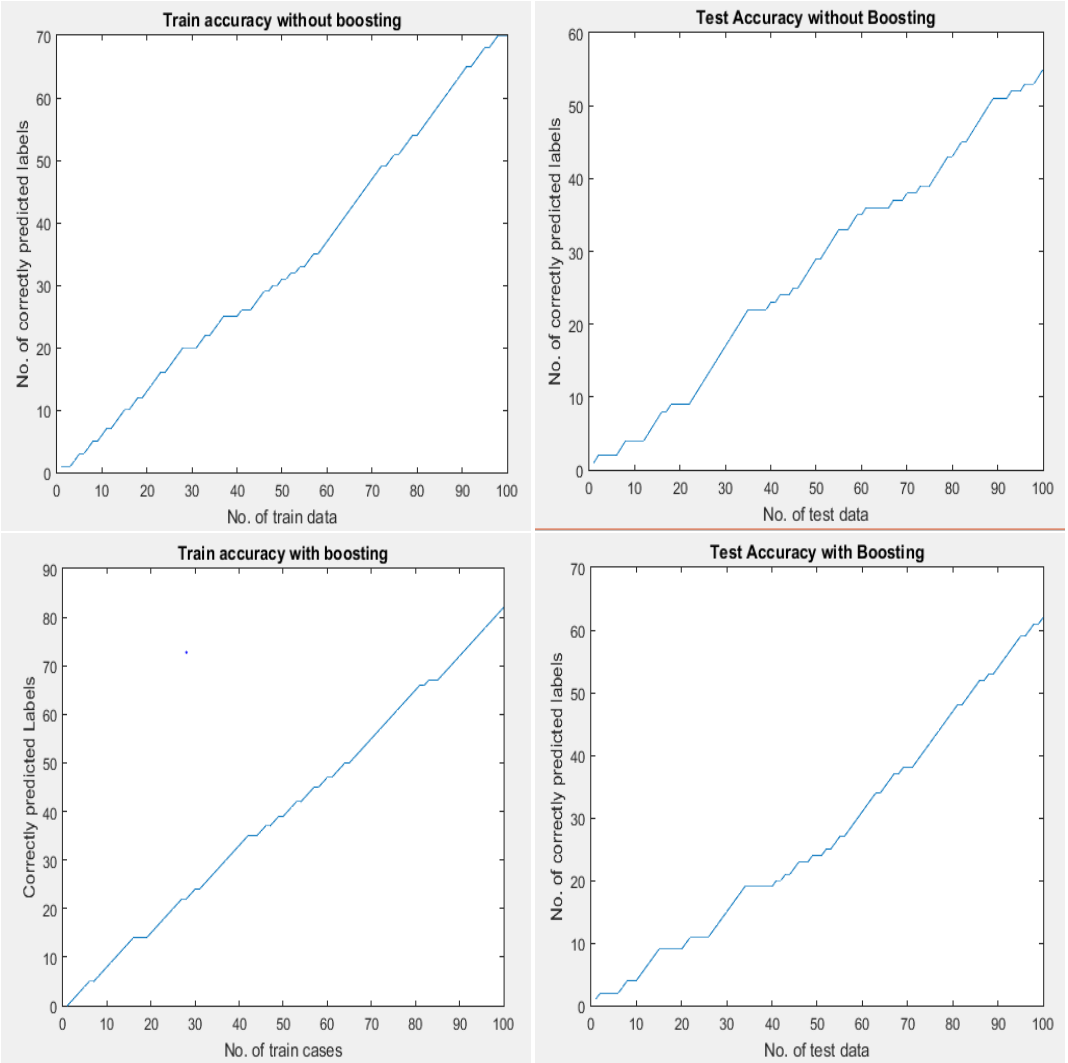
Neural Networks

The following results were obtained for neural networks:
These were the plots obtained by our NN model when run on the data. The trained neural network was a 3 layer deep neural network and there were 10 neurons in each layer. We run our network on the remaining data after training on first 2000 points and tested in on the remaining data.



SVM(Combined Model)

While the data was trained on 500 data point, only last 100 points were taken to measure train accuracy. The reason is, we designed the model in such a way, so that it would be penalized more on more recent data. So accuracy of predictions for earlier train data would be low, but it won't matter anyway. We are concerned with the predictions on last 100 train data points only.This is the reason why we took test data to be only 100 values which were immediately after train data, as the accuracy will decrease as me move further away from train data points.



Comparison with individual models

When we train individual models, we get a max accuracy of 61% for TDNN and of 55% for SVM. These are the results obtained by running our individual NN model:

Delay	Configuration of Layers	Number of Neurons	Accuracy
5	5-5-5	15	57.41
10	10-10-10	30	55.81
10	5-5	10	61.76
12	7-7	14	57.2
12	12-12	24	55.51

7 Discussions

While doing the SVM classification for test data we used current day's value along with the previous day's values as feature vectors. While the current day's values were available for train data, they won't be available during test run. So the current day's values were generated from NN algorithm. But it will cause our test data to have errors, as NN will not predict the values with 100% accuracy. But the results are better than modelling using just the previous data. Even though the new features from NN have errors they are providing additional information to the model which causes an increase in overall accuracy. While the results don't improve very much by just using a combined model, they do improve significantly when we also use Ensemble methods.

For our model we used Adaboost boosting method which increased the accuracy of our model by about 7-10%. While implementing boosting at first, the train accuracy almost reached up to 90% by increasing the number of rounds, which can be expected as boosting works to increase the train accuracy. But this also meant that our model was over fitting. So in consequent train and test runs, we varied the no. of times we boosted our algorithm, to find an optimal test accuracy. So the Test accuracy also improved by about 7%.

So finally we were able to achieve an accuracy of 62% over test data, which can be considered on the higher side comparing it with the accuracies of currently known open-source models. While it is speculated that some high profile Investment Banks and Wall-Street firms have developed models which are far more accurate, amongst the models that we know of our model's performance can be judged as pretty good.

Acknowledgments

We would like to express our gratitude to our guide, Professor Piyush Rai under whose guidance we worked on this project on the topic Financial Time Series Prediction which helped us in understanding Machine Learning and gave us an opportunity to learn about a variety of new concepts.

References

[1] Mahdi Pakdaman Naeini, H. Taremian and Homa Baradaran Hashemi,*"Stock market value prediction using neural networks"*, CISIM, 2010 International Conference on, Krackow, 2010, pp. 132-136.

[2] Wei Huang,, Yoshiteru Nakamoria, Shou-Yang Wang.*"Forecasting stock market movement direction with support vector machine*, Computers and Operations Research, Volume 32 Issue 10, October 2005, Pages 2513-2522

[3]Fang Yixian, Wang Baowen, Wang Yongmao, *The stock index forecast based on dynamic recurrent neural network trained with GA"*, The 20^{th} Pacific Asia Conference on Language,Information and Computation, 2006-11