Stock Market Prediction Using Machine Learning

Divya Agarwal
College of Computer and Information Science
Northeastern University
agarwal.d@husky.neu.edu

Abstract

The goal of this project is to understand and implement the functioning of Artificial Neural Networks, while studying the stock prices in time and trying to predict the future stock prices. We consider the possibility of using feed forward neural network, recurrent neural network and LSTM. We predict the stock prices training our model on historical data, dated sixteen years back data using LSTM networks and the test accuracy obtained in the process is 99.65%.

Introduction

Maintaining records of traded commodities go back to thousands of years. Merchants used to keep a record of traded goods to try and predict price trends so that they could benefit from it. In finance, quantitative analysis dates to early 1980s. Stock market analysis has been quite a popular area of interest for both researchers and stock market firms. The randomness and unpredictably of the market makes it a challenging task to predict the values accurately. In stock market, quantitative analysis is the study of how certain variables correlate with stock price behavior. A lot of academic research papers have also been published to predict the stock prices using neural nets with varying degree of success. Until recently, only academicians who knew algorithms could build these models. But now with machine learning architectures like tensor flow anybody can build powerful predictive models trained on massive data sets.

The objective of this study is to implement artificial neural network to predict the closing price for any given date after training on NASDAQ stocks. We will also discuss various types of neural networks and why we choose Long Short Term Memory network for our study. We will train the model on historical database of companies, like Apple, Google, Microsoft, JPMorgan and so on from the year 2000 to the current system date. We used 85% for training the model and the rest 15% for validation. In the upcoming sections, we will discuss how to implement the machine learning algorithms to obtain high accuracy results. Also, we will discuss the features used to make the predictions and how they affect the stock prices.

Background

Dataset

As discussed earlier we will use historical data sets of 16 years (01/01/1990 to Current System Date) from yahoo finance obtained for companies like Yahoo, Google, Microsoft and Apple to train and test our model. Each of these companies have the following columns in the historical dataset. [14]

1. Date (YY-MM-DD)
2. Open (float)
3. High (float)
4. Low (float)
5. Close (float)
6. Volume (float)
7. Adjclose (float)

We will filter out open, close and high values from the obtained data. This is a series of data points indexed in time order or a time series. Our goal will be to predict the closing price of any given data and company after training. We will use data normalization to normalize the data before training the model. Normalizing the data improves convergence. We will use the following equation to normalize each data point,

$$n_i = \frac{p_i}{p_0} - 1$$

Here, p_i represents the current price, divided by initial price, p_0 and subtracting one to get percentage change. We will de-normalize the data after it starts making predictions using, to get real world value,

$$p_i = p_0 + 1$$

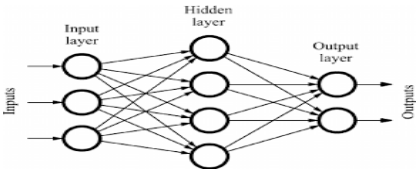
Feed Forward Neural Network

A feedforward neural network is an artificial neural network wherein connections between the units do not form a cycle. The feedforward neural network takes an input data and transforms it to give output results. For example, if we want to label the images into categories say “cat” or “dog”, we will train the feedforward neural network with a sample

dataset to minimize the errors while doing the categorization.

The feedforward network does not consider the previous results into account to make the perditions. It works only with the current information available. That's the reason we cannot use it to make prediction when memory matters or for a sequence of data. For example, to predict the next frame in a movie we cannot use feed forward neural networks since that would require a sequence of image vectors as input not just a single image. Since the probability of next frame occurring will depend on the probability of the frame before it. [20]

Figure 1. Feed Forward Network.

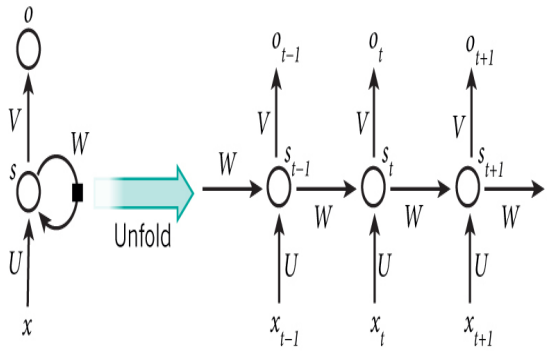


Recurrent Neural Networks

Recurrent neural network is an artificial neural network which allows directed cycle connections to exist among its units. It is especially useful for sequential data where each unit contains the information about its previous input. It allows network to learn from its previous predictions and new inputs.

We can think of RNNs as units having memory which captures knowledge that has been discovered so far. Theoretically, RNNs can make use of information in arbitrarily long sequences, but in real time they have limitations and can look back only a few steps. Figure 2 shows what a typical RNN looks like.

Figure 2. Unfolding RNN in order of time.



The figure 2 shows the unfolding of an RNN into a full network. Here, x_t is the input at time step *t*. s_t is the hidden state in the layer *t*. It is calculated based on the previous state s_t-1 and new input at current state x_t . The RNNs memory carry forward can be describes mathematically as:

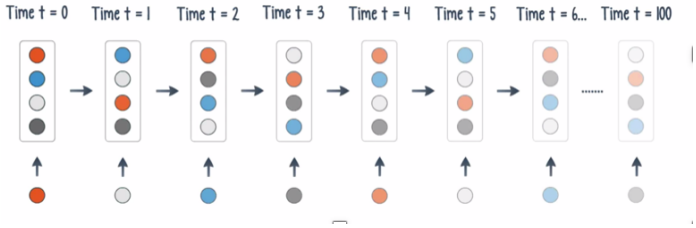
$$s_t = f(Ux_t + Ws_{t-1})$$

The hidden state at time step t, is s_t. It is a function of the input at the same time step x_t, modified by a transition matrix, U added to the hidden state of the previous time step t-1, s_t-1 multiplied by its own weight matrix W. The weight matrix accord the importance to the present input state and previous hidden state. Thus, acting as filters. Using backpropagation, the error generated is returned into the layer until it cannot be reduced any lower. The function *f,* acts as a squashes the sum generated by the input sate and hidden state. Since, this feedback loop occurs at every time step in the series, each hidden state contains traces not only of the previous hidden state, but also of all those that preceded s_t -1, for as long as memory can persist. [18]

Vanishing Gradient

The large increase in norm of the gradient during training of neural nets is referred to as exploding gradients. This event occurs because the long-term components, which can grow exponentially more compared to short term component. The vanishing gradient is the reverse. The long-term components tend to norm 0 exponentially, making it impossible for the model to learn correlation between temporally distant events.

Figure 3. Vanishing Gradient Problem.



Related Work

Logistic Regression

Logistic regression is more probability based classification. Regression means, we try to fit the existing linear model to the available feature space. [16].

Logistic regression is preferred in case the response variable can take only binary values (yes or no). It is used to produce a categorical outcome. In stock price prediction, the trend of next day's price can be categorized into two classes, which is whether the price of the stock is increasing or decreasing hence logistic regression can be used. [5]
It is used for predicting a binary outcome or used to categorize the data.

Figure 4. Mathematical representation of logistic regression.

$$p = \frac{\exp(c_0 + c_1x_1 + c_2x_2 + \cdots + c_kx_k)}{1 + \exp(c_0 + c_1x_1 + c_2x_2 + \cdots + c_kx_k)}$$

Prediction of stock prices is a continuous outcome, since the input is sequence of data. Thus, logical regression can be used only if we want to foresee an increase or decrease. It cannot be used to make mathematical or real time prediction of the market. [9]

Artificial neural networks always outperform logistic regression model because of their dynamic nature, which matches the noisy stock prices data therefore we will not be exploring logistic regression in our paper.

Decision Trees

Decision trees are tree-like graph in which each internal node represents a test on the attribute and each outgoing edge represents some possible value of that test. The tree like structure is formed due to splitting of dataset based on the rule that maximizes the separation of the data. The leaf nodes represent the multiple possible classes. Decision trees are used for classification problems most of the times. The cost of using a decision tree to forecast is log(n) where n is the number of data points used to train the algorithm. [8]

The figure 5 depicts the working of decision tree algorithm on stock market analysis. The tests are established on each node based on different attributes. Then the data is classified and a label is assigned in the leaf node.

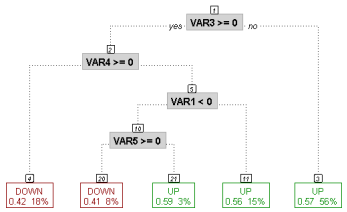


Figure 5. Example of Decision tree for stock market analysis.

The min sample parameter decides the number of minimum samples the decision tree should split a node in.

Decision trees sometimes fail in generalizing the data correctly due to complex tree structures that are formed during the splitting process. This results in over fitting or under fitting of the data. This can be a major disadvantage of using this algorithm. Spitting of continuous variables implicitly may cause loss of necessary information, which is another drawback. Below in Figure 6, is the general algorithm for decision tree. [9]

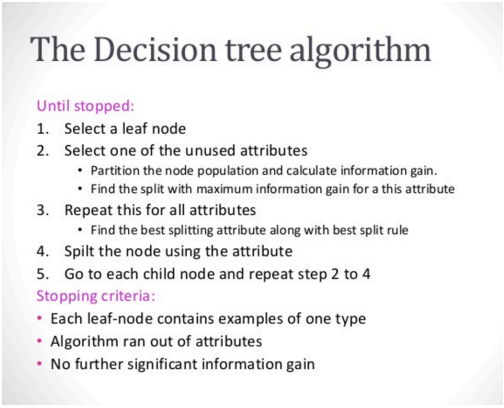


Figure 6. General Decision tree algorithm.

Project Description

Long Short Term Memory Networks

The Long short term memory network or LSTM network, is a modified Recurrent neural network that is trained using backpropagation through a series of time and eliminates the problem of vanishing gradient.

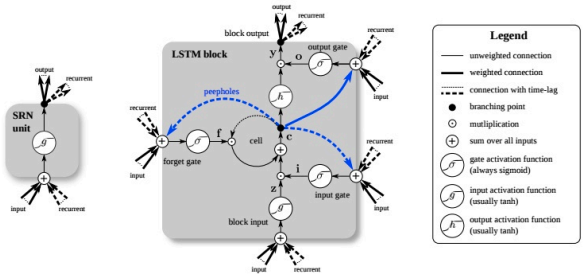
It addresses the difficult sequence problems in Machine learning and can be used to create large recurrent neural nets.

LSTM networks have memory cells instead of neurons, that are connected throughout the layers.

A block consists of gates, to manage the blocks state and output. Every block operates upon an input sequence and each gate uses the sigmoid activation units to control if they get triggered out. Thus, making the change of state and new information flowing through the block conditional.

There are three kinds of gates within a unit. Firstly, the Forget Gate, it conditionally decides what information to throw away from the block. Next, Input Gate, based on a condition decides that value from the input which will update the memory state. Finally, Output Gate, conditionally decides what to output based on input and the memory of the block.

Figure 4. Simple Recurrent Network (left) and LSTM block as used in a hidden layer of RNN.



We will be building our own LSTM network using keras with tensor flow backend. For our training data, we will be using data from yahoo finance. We will perform a time series analysis on the data dated from 1st January 1990 to current day,2017.

We will create the URL dynamically by keeping the end date and the company name as variables. So that we can change the size of data and the company to train our model. We will create a model to predict closing price of any given date after training. We will fetch the data from the internet into a csv file. Then we will load the data using a custom load function to read the csv data and normalize it to create a transformation matrix.

Rather than feeding the values directly into the model normalization ensures convergence faster. To build the model we will first initialize it as a sequential model or a sequence of inputs. Since it will be a linear stack of layers. Then we add our layers to the model. The first layer is a long short term memory layer. Since we need to allow the information to persist to make accurate predictions we use LSTM network. Recurrent networks can accept a sequence of vectors as input.

Recall that a feed forward neural network, the hidden layer weights are based only on the input data. But in a recurrent net, the hidden layer is a combination of the input data at every time step (i.e. the current time step and the previous time step). Since the feedback loop is occurring at every time step in the series, each hidden state has traces of not only the previous hidden state but of all those that preceded it. That is, they can relate pervious data with the present task. In the long short term memory network, we replace the neurons of recurrent neural networks, with memory cells to eliminate the vanishing gradient problem.

Solving Vanishing Gradient Problem

Assume we have a hidden state s_t at time step *t*. By simplifying things, by eliminating biases and inputs, we get.

$$s_t = \sigma(W s_{t-1})$$

Then we can say that,

$$\frac{\partial s_{t'}}{\partial s_t} = \prod_{k=1}^{t'-t} W\sigma'(W s_{t'-k}) = \underbrace{W^{t'-t}}_{\%\%} \prod_{k=1}^{t'-t} \sigma'(W s_{t'-k})$$

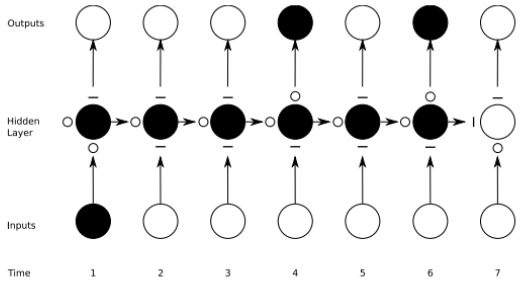
The factor highlighted with %% is the vital one. If the weight is not equal to 1, it will either decay to zero exponentially fast in $t'-t$, or grow exponentially fast. [4]

In LSTMs, we have the state s_t. The derivative here is of the form,

$$\frac{\partial s_{t'}}{\partial s_t} = \prod_{k=1}^{t'-t} \sigma'(v_{t+k})$$

Here v_t is the input to the forget gate. There is no exponentially fast decaying factor involved. Consequently, there is at least one path where the gradient does not vanish. For the complete derivation, see [11].

Figure 5. Preservation of gradient information by LSTM. As in the figure the shading of the nodes indicates their sensitivity to the inputs at time one; in this case, black nodes are maximally sensitive to the inputs and white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open ('O') or closed ('-'). The memory cell remembers the first input if the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell. [20]



Experiments

Feature Selection

The following are the features we tried while building our model and appended to the list gradually, to increase the accuracy of the model.

- *f1*: Opening Price(dj) – Opening Price(dj_{-1})
- *f2*: Closing Price(dj) – Closing Price(dj_{-1})
- *f3*: High(dj) - High(dj_{-1})
- *f4*: Low(dj) - Low(dj_{-1})
- *f5*: Volume
- *f6*: Dividend
- *f7*: Yield
- *f8*: Opening / (difference of high, low)
- *f9*: Closing / (difference of high, low)

Each of the features are explained below in brief.

- *f1* and *f2* are the difference of opening or closing stock price of the given day and the previous day respectively.
- *f3* and *f4* is the difference of highest or lowest stock price of the given day and the previous day respectively.
- *f5,* is the volume that represents the amount of stocks that have been traded on that day. This depicts the interests in buying or selling a company's shares.

- *f6,* the dividend you will receive per year per share of a stock.
- *f7,* yield gives the percentage return on the dividend.
- *f8* and *f9,* ratio of opening or closing stock price for that day to the difference of highest and lowest price for stock on the same day.

Result Discussion

Stock market prediction for a given company was implemented using the features mentioned above. All algorithms mentioned above are implemented with the help of keras library[number] with tensorflow backend. We divided the dataset into 85% for training and 15% data for testing the implemented model.

The artificial neural networks use hidden states or layers which can be modeled mathematically as. [19]

$$h_t = \phi(ws_t - Uh_{t-1})$$

The hidden state at a given time step *t*, is the function of the input at the same time step modified by a weight matrix added to the hidden state of the previous time step multiplied by its own hidden state to hidden state matrix.

The code snippet in fig 6. shows building of LSTM networks model using keras with tensorflow backend [5]. The model is a sequential model. As the first LSTM layer, we set the input_dimenssion to 1 and output to 50 units. Setting return_sequence to true means that this layer's output is always fed into the next layer.

```
In [4]: #Step 2 Build Model
model = Sequential()

model.add(LSTM(
    input_dim=1,
    output_dim=50,
    return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(
    100,
    return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(
    output_dim=1))
model.add(Activation('linear'))

start = time.time()
model.compile(loss='mse', optimizer='rmsprop')
print 'compilation time : ', time.time() - start

compilation time :  0.0409510135651
```

Figure 6. Code snippet for building an LSTM network.

We add 20% dropout to the first layer and initialize the second layer as another LSTM with 100 units and set return_sequence to false, since its output is only fed into the next layer at the end of the sequence.

It doesn't output a prediction for the given sequence, instead it gives a prediction vector for the whole input sequence. Then we use the linear dense layer to aggregate the data from this prediction vector into one single value. Then we compile the model using mean squared error loss function and use gradient descent as the optimizer. Next, we train the model using the fit function [5].

Then we can test it to see the results and plot it on a graph for the next n=50 steps. The following are the results of plotting the graphs for yahoo finance data collected for a period of 5840 days, which covers a period of sixteen years starting 1 January 2000 to Current System Date (last test performed: 17 April 2017). The data was split into 4964 records for training set and 876 records for testing set. The accuracy obtained is around 99.75% for the predictions.

Figure 7. Code Snippet for training the model.

```
In [5]: #Step 3 Train the model
model.fit(
    X_train,
    y_train,
    batch_size=512,
    nb_epoch=1,
    validation_split=0.05)

Train on 3523 samples, validate on 186 samples
Epoch 1/1
3523/3523 [==============================] - 8s - loss: 0.0098 - val_loss: 0.0011
Out[5]: <keras.callbacks.History at 0x10f8b9b90>
```

Figure 8. Plotting of Actual vs Predicted stock prices for Apple.

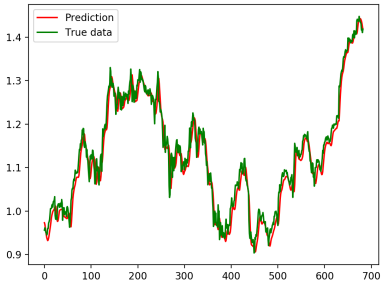


Figure 9. Plotting of Actual vs Predicted stock prices for Yahoo

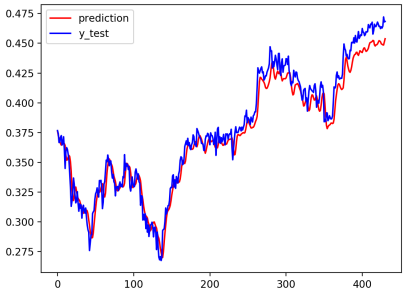


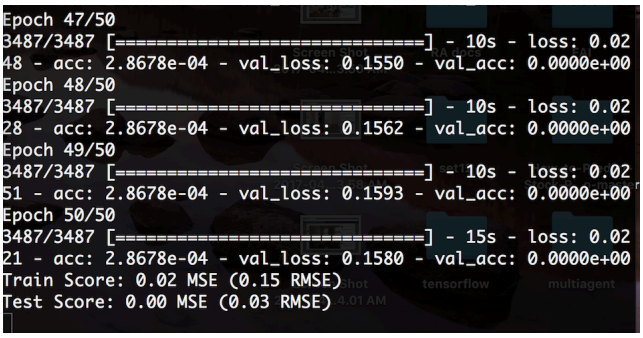
Figure 10. Table showing the obtained values.

Number of Epochs	Training Score (percentage)	Testing Score (percentage)
40	96.73	99.54
85	97.45	99.67
140	97.86	98.78

Figure 11. Scores for prediction of Apple Stock price.

Number of Epochs	Training Score (MSE)	Testing Score (MSE)
50	0.02	0.00
150	0.00	0.00

Figure 12. Demo Run for Apple Stock price forecasting.



Conclusion

In this paper, we discussed the application of different machine learning algorithms to predict stock market prices for various companies. We described the theory behind feed forward neural networks, logistic regression, recurrent neural networks, long short term neural networks and decision tree algorithm. The stock market depends on a lot of varying factors, collecting all the factors and applying it to a machine learning model is a tedious task. But we can achieve a high accuracy score by training our model with some of the most important features that determine the rise or fall. Thus, we can use machine learning algorithms to predict the trend of stock market accurately which is otherwise considered unpredictable. This project can be further extended to improve the performance by using more data mining techniques.

We have explored the best in recurrent neural networks to predict stock prices in our research, future researchers may include more methods for finding the best model for predicting stock prices. We could use Support Vector Machine to forecast stock prices and compare the working of artificial neural networks compared to support vector machines. Building a real-time application to automate the buying and selling of stocks by applying our forecasting model, is another interesting idea to explore. Finally, the prediction model we built explores very limited features. Many more features could to added to the existing features vector list to improve accuracy and make it closer to realistic software. Other trading strategies may also be investigated.

References

[1] On the difficulty of training recurrent neural networks, Razvan Pascanu, Tomas Mikolov, Yoshua Bengio.

[2] Recurrent neural networks, Introduction to RNNs, Denny Britz.

[3] A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index, Alaa F. Sheta, Sara Elsir M. Ahmed, Hossam Faris.

[4] Learning Sequence Representations, Bayer, Justin Simon.

[5] https://keras.io

[6] Machine Learning in Stock Price Trend Forecasting, Yuqing Dai, Yuning Zhang.

[7] Forecasting stock market trends by logistic regression and regression and neural networks, Makram Zaidi, Amina Amirat.

[8] Forecasting stock market trends by logistic regression and neural networks evidence from KSA Stock Market, Makram Zai.

[9] Stock Market Prediction Using Machine Learning, Sankar Gireesan Nair, Veera Venkata Sasanka Uppu.

[10] On the difficulty of training recurrent neural networks, Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio.

[11] Prediction of stock prices using analytical techniques, Carol Hargreaves, Yi Hao.

[12] Stock Market Prediction Using Artificial Neural Network, Prakash Ramani, Dr. P. D. Murarka, in International Journal of Advanced Research in Computer Science and Software Engineering.

[13] https://finance.yahoo.com/

[14] http://www.nasdaq.com/

[15] https://www.tensorflow.org/

[16] http://pandas.pydata.org/

[17] https://matplotlib.org/

[18] https://deeplearning4j.org/lstm.html

[19] Siraj Raval, Deep Learning -7, Coding Challenge.

[20] https://en.wikipedia.org/wiki/Artificial_neural_network