Stock Prediction with Machine Learning

Antony Alexos AALEXOS@UTH.GR
Antony Evmorfopoulos AEVMORFOP@UTH.GR
Tilemachos Tsiapras TTSIAPRAS@UTH.GR

Abstract

A good approach to predict future price of a stock, based on its past price and other factors was investigated. In this paper we try to predict whether the price of a stock will go up or down on the next day, based on a classification model that we built, using LSTMs from keras library.

1. Introduction

Predicting stock and stock price index is difficult due to uncertainties involved. There are two types of analysis which investors perform before investing in a stock. First is the fundamental analysis. In this, investors looks at intrinsic value of stocks, performance of the industry and economy, political climate etc. to decide whether to invest or not. On the other hand, technical analysis is the evaluation of stocks by means of studying statistics generated by market activity, such as past prices and volumes. Technical analysts do not attempt to measure a security s intrinsic value but instead use stock charts to identify patterns and trends that may suggest how a stock will behave in the future. Efficient market hypothesis by experts, states that prices of stocks are informationally efficient which means that it is possible to predict stock prices based on the trading data. This is quite logical as many uncertain factors like political scenario of country, public image of the company will start reflecting in the stock prices. So, if the information obtained from stock prices is preprocessed efficiently and appropriate algorithms are applied then trend of stock or stock price index may be predicted.

Since years, many techniques have been developed to predict stock trends. Initially classical regression methods were used to predict stock trends. Since stock data can be categorized as non-stationary time series data, non-linear machine learning techniques have also been used. Artificial Neural Networks (ANN) is a machine learning algorithms which is most widely used for predicting stock and stock price index movement. ANN has its own way to learn patterns. ANN emulates functioning of our brain to learn by creating network of neurons. This study focuses on a modification of ANN, a Long Short-Term Memory Neural Network, which is an optimization of RNNs. Here we use the LSTMs(Hochreiter & Schmidhuber, 1997) in a classification approach, in order to predict the next day, if the stock's price will be up or down.

University of Pennsylvania, CIS 419/519 Course Project.

2. Dataset and Features

The basic data that we used are downloaded from Yahoo through its API and it is the price of Goldman Sachs stock. The dates that we chose are from 31/12/2014 to 19/3/2019, and apart from that we also use the price of Bank of America, Barclays, Credit Suisse, JPMorgan, Morgan Stanley, NASDAQ, Hang Seng Index, NYSE, Nikkei 225 and VIX. We also used the technical indicators Moving Average 7 and 21, Exponential Moving Average(EMA), Moving Average Convergence Divergence(MACD), Bollinger Bands, Momentum and Log Momentum. The last additions to our dataset are three fourier transformations. Because of the Bollinger Bands we started the dataset 29 days after the starting data of the data, because until day 29, some of the data are missing. The idea of adding all of this data is that the price of a stock is affected by a majority of factors; so the more we add, the better prediction we may have.

3. Methods

3.1. Long Short Term Memory

Long Short Term Memory networks(Wei Bao, 2017) are neural networks that are being used for solving time series problems. As such, they provide the best possible solution in solving the problem of predicting the prices in stock market. This kind of network is used to recognise patterns when past results have influence on the present result. For this reason, our experiments and efforts were revolved around LSTM networks.

Our first network was simple LSTM network of 50 neurons, with one hidden layer. We tried different type of models with this kind of architecture. Firstly, we passed

our data through the features dimension of the LSTM cell, then through the timestep dimension. Another model we built was making the LSTM network stateful - with memory between the batches of data we passed, making it learn from every sequence instead of every epoch. Afterwards, we applied dropout of 0.2 to the model that gave the best results. That was done in order to avoid any overfitting of our algorithm to our training set, trying to make it generalize better to unseen data. All these models were built in order to fully capture the nature of our problem, trying to see which algorithm would fit best as a solution.

The way we evaluated our model was through a custom metric, named gain. While classic metrics, like mean squared error, mean absolute error etc, can be used to evaluate how close is our prediction to the actual price of the stock, the problem itself suggests that we should see if the model is gaining money or not. With that in our mind, if we predict that the stock will move to the same direction as the actual move, then we won the absolute value of the subtraction of the closing price from the opening price. In the same way we lost money if we predict the opposite direction from the markets.

Afterwards, we built stacked LSTM models, trying to take advantage of the memory the LSTM cells provide to a network. The architecture is composed from two layers, one with 50 neurons followed by one with 30. We followed the same principle as before while building our models.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 50)	16000
lstm_1 (LSTM)	(None, 30)	9720
dense (Dense)	(None, 1)	31

3.2. XGBoost

We used XGBoost in order to see the importance of our features(Dey et al., 2016). From there, we could implement a different model which hypothetically be better since we would use only the data that capture better the information we want to learn, however no such result was achieved since only 2 out of 30 features were considered as not informative enough. For the record, XGBoost has been a proven model in data science for its speed and accuracy. In XGBoost the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals.

3.3. PCA

In order to reduce the dimensionality of our data, we implemented principal components analysis(PCA)(Jolliffe & Cadima, 2016). PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors (each being a linear combination of the variables and containing n observations) are an uncorrelated orthogonal basis set. From this analysis we resulted to 5 variables, which try to capture the most information from our data and we followed the same procedure, to build the new models, as before.

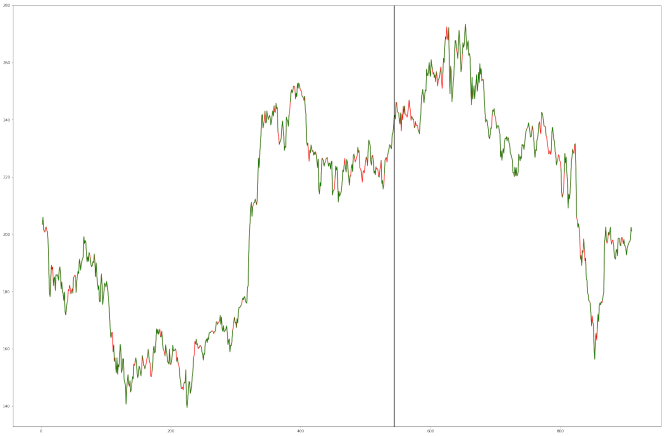
However, in the results of this process, it can be noticed, that even though it wasn't possible to predict the stock price, the predictions were following correctly the direction the price was going. For this reason , we transformed the whole problem to a new one. Instead of predicting the price directly, we will try now to predict the direction of the price where the price is heading, making it a classification problem.We transform our data to binary, 0 if the price fell that day, 1 if the price went up. That way the information our models need to process is much simpler and therefore, much easier to learn.

4. Results and Discussion

Time series data, as the name indicates, differ from other types of data in the sense that the temporal aspect is important. On a positive note, this gives us additional information that can be used when building our machine learning model, that not only the input features contain useful information, but also the changes in input/output over time. However, while the time component adds additional information, it also makes time series problems more difficult to handle compared to many other prediction tasks.

So the gain metric in the classification approach has given us a gain of 818.18, which is a very impressive result, because at the time of testing the LSTM in the data the Goldman Sachs stock only gains 38 points. We can observe that our algorithm surpasses that by a lot.

The accuracy of the algorithm is 0.736986. This is the percentage that we predicted right the days that the stock went either up or down. The accuracy can also be seen from the graph below where we see with green color the days that we predicted right, and with red color the days that we predicted wrong.



5. Conclusion and Future Work

In conclusion with this work, we have achieved a really good result. The accuracy of the algorithm may be only 0.736986, but this is above 51% which is considered satisfying by most professionals. But accuracy combined with the gain metric, which is quite high doesn't mean that we have completely solved the stock market prediction problem. We have some future plans to make this algorithm more reliable and more successful. In the future we will add more complex and better algorithms like GANs, we will also add prices targets and stop losses to achieve a better result, and finally we will implement the algorithm in MQL4 in order to do live trading with it.

References

Dey, Shubharthi, Kumar, Yash, Saha, Snehanshu, and Basak, Suryoday. Forecasting to classification: Predicting the direction of stock market price using xtreme gradient boosting, 10 2016.

Hochreiter, Seep and Schmidhuber, Jurgen. Long short-term memory. *NEURAL COMPUTATION*, 9(8):1735–1780, 1997.

Jolliffe, Ian T. and Cadima, Jorge. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065), 04 2016.

Wei Bao, Jun Yue, Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):1–24, 07 2017.