Machine Learning Engineer Nanodegree

Capstone Proposal: Build a Stock Price Indicator

March 10th, 2018

Michael Løiten

Proposal

Domain Background

Stock market trading has existed since 1602. From its very start the stocks markets has not only been a place for passive trading between buyer and seller, but it has also influenced local and global economies.

Although stock trading traditionally has been made manually by man, increased computational power have paved way foralgorithmic trading which typically builds on mathematical models or statistical techniques, and has been used extensively since the 1970s.

Traditionally statistical techniques such as looking at Bollinger bands have been used for making systematic trading decisions. However, in today's marked, more information of the system is usually needed in order to reach good investment strategies.

One way to add new information is to use regression methods through machine learning, to try to predict the future value of a stock. In addition, techniques such as those found in reinforced learning can help traders (human and machines) make better decisions based on more data than humans do (which often leads to short term decisions making the market volatile).

The idea of using machine learning for trading is far from new, and both academics and students have been studying the topic.

Apart from finance being interesting to studying in its own, in the domain of machine learning it presents an interesting problem due to the nature of data, being sequential. This means that traditional training techniques such as randomization of the training data and cross validation techniques makes no sense. The training must also be considered perishable as the stock market is rapidly changing.

In addition, financial data is known for being non-stationary stochastic (the stochastic nature is in some cases be modelled as a Brownian motion Markovian process). There are several tests which can show that the stock prices are non-stationary. As the time series are non-stationary, the statistical moments are dependent on time, and clever ways of scaling the data is needed.

The techniques used to deal with stock data is also transferable to other sequential data such as weather forecasting, power distribution on the electrical grid, maintenance prediction, and other fields where sensory data is of importance.

Problem Statement

In this project, we will build a stock price predictor for the 50 stocks with the highest weights in the Standard and Poor's 500 (S&P500) portfolio as of 2018-03-06. The predictor will be trained on historical daily data including the opening price (Open), the highest price the stock was traded at (High), the lowest price the stock was traded at (Low), how many stocks were traded (Volume) and the closing price adjusted for stock split and dividends (Adjusted Close (see for example "02-07 Dealing with data" in Udacity's course Machine learning for Trading for more info)). The predictor should be able to predict the Adjusted Close up until 28 days into the future. And the predicted stock value 7 days from the date of prediction should be within 5% of actual value, on average.

Although interesting, trading decisions based on machine learning will be outside of scope in this project. Also, automatic updating of the stock data will be outside the scope of this project.

Datasets and Inputs

For the scope of this project, we will use daily stock data containing open, high, low, close, volume (ohlcv) together with the adjusted close for the 50 stocks with the highest weights in the S&P500 portfolio together with ^GSPC itself.

The weights for S&P500 has been collected from SlickCharts the 6th of March 2018.

The daily stock data have traditionally been readily available through free APIs such as Yahoo! Finance, Google finance and EDGAR. However, late 2017, several APIs got deprecated. Reading from pandas-datareader (a continuation of the now deprecated `pandas.io`)

> Yahoo! Finance has been immediately deprecated. Yahoo! substantially altered their API in late 2017 and the csv endpoint was retired.

> Google'a API has become less reliable during 2017. While the google datareader often works as expected, it is not uncommon to experience a range of errors when attempting to read data, especially in bulk.

As good free alternatives, Alpha Vantage and Quandl remains.

All stock data is downloaded through Quandl, with the exception of the Standard and Poor 500 index, which is manually downloaded from Yahoo! Finance. (Although Yahoo! Finance has discontinued their free API, workarounds like`fix_yahoo_finance` exists, which uses a smart way to retrieve the needed breadcrumbs. Although the download could be through a script, the solution can render obsolete in the near future.)
The scope is set to look at a fixed dataset. A possible extension would be to update and train the dataset daily. The most stable way to retrieve the data would probably be to buy the data from a professional vendor, for example through Quandl.

Solution Statement

A training model will be build. The input for this model is a start date, an end date (not earlier than 28 days prior to the prediction) and together with a list of ticker symbols from the 50 symbols of the S&P500 set. The output will be the model used for prediction.

A data cleaning module based on the 01-05 Incomplete data of Udacity's course Machine learning for Trading will be made. The data will pass through this module before the final prediction.

The predictor module will be based on the model, taking one or more of the available ticker symbols from S&P500 (like GOOG, AAPL) as an input and yield the predicted adjusted close prognosis.

In the end, these modules will be made callable through a driver script.

Benchmark Model

Although other models are freely available online (see for example the blogs in the Additional resources section), the model will be benchmarked against simple models of the adjusted close price, like

- Prediction equal the latest day
- Ordinary least square
- Polynomial regression of order 5

- Random guessing using a Gaussian distribution with the mean equaling the last day before prediction, and the standard deviation being based on the historical standard deviation.

Evaluation Metrics

The relative error of the adjusted close will denote the success of the model. The relative error is defined as

```
|True value - Approximated vale|/True value
```

Project Design

In order to successfully build a model, one need to get a feeling of the data set. First the quality of the data will be checked. Are there a lot of missing data? Can all the missing data be mended, or should some ticker symbols be left out? A visualization of the different data attributes will be done to get a feeling of general trends and patterns. Feature engineering will also be needed as we are working with categorical data. The features will be created by shifting the data as explainedhere.

Following, the simple benchmark tests will be performed in order to see if fancy models obtained through machine learning is needed at all. If it turns out that some simple methods performs really well, it could be interesting to see if machine learning has something to add to the problem at all.

The next step in the project would be to experiment with different techniques. This can potentially take a lot of effort as there are many techniques to explore. However, the following techniques can be interesting to have a look at

- KNN, as this is a simple model, and has proved successful at for example QuantDesk
- Generalized Linear Models like Bayesian Regression.
- Neural nets, in particular convolutional neural networks as these have a tendency to perform well when the "relative positions" of the features is important.

When checking evaluating the models we will look at

- How the performance of the models scales with the amount of data points
- The learning curve for variance and bias.
- The model complexity
- The amount of features, i.e. how far back would we need to look in order to obtain reasonable results.

Finally, the full pipeline of training, cleaning and predicting will be made.

Additional resources

- Udacity's course Machine learning for Trading
- Machine Learning Strategies for Time Series Forecasting
- Machine data for sequential data: A review
- The Machine learning mastery time series tutorial
- The Medium blog (note that this blog does a good job explaining, although the results are somewhat poor).