Machine Learning Engineer Nanodegree

Capstone Proposal

Miguel Tasende
January 17th, 2017

Proposal

Domain Background

Predicting the stock price trend by interpreting the seemly chaotic market data has always been an attractive topic to both investors and researchers. Among those popular methods that have been employed, Machine Learning techniques are very popular due to the capacity of identifying stock trend from massive amounts of data that capture the underlying stock price dynamics. According to market efficiency theory, US stock market is semi-strong efficient market, which means all public information is calculated into a stock's current share price, meaning that neither fundamental nor technical analysis can be used to achieve superior gains in a short-term (a day or a week) [Dai[1]].

Much economic research has been conducted into the Efficient Markets Hypothesis theory, which posits that stock prices already reflect all available information and are therefore unpredictable. According to the EMH, stock prices will only respond to new information and so will follow a random walk. If they only respond to new information, they cannot be predicted. Most research with machine learning forecasting has focused on Artificial Neural Networks (ANN). Recent research in the field has used another technique known as Support Vector Machinesin addition to or as an alternative to ANNs.[Madge[2]]

In this Capstone Project, some other techniques will be explored, to try defy the Efficient Markets Hypothesis.

Motivation: At some moment I thought the goal of science was to make good predictions of natural phenomena. Now I believe that philosophically, the ultimate goal of science is to learn how to make good decisions. When the latest technologies are taken into account, the "feedback loop" (problem-measurements-models-predictions-decisions-results-more problems) gets everytime simpler and happens faster. That blurs the line between "science" and "business". In the recent past, many scientists and engineers have changed their careers to the finance sector, which may confirm that blurry line. I believe, in the future, the finance sector, and business in general, have much to offer to science, and hybrid organizations could benefit both areas, a lot. In particular I hope the differentiation between "applied science" and "basic science" will become more and more irrelevant as the "applications" become more subtle and complex (enough to include very abstract thinking), and the "basic knowledge" finds better criteria for deciding the ultimate value of a theory. I believe that can be done without simplifying basic science and, at the same time, without adding unnecessary time delays and burdens to applied science and business. It seems to me, that is the path that technology is opening.

Problem Statement

Given a time series of (Open,High,Low,Close,Volume,Adjusted Close) values for the stocks that are presently (as of January 17th, 2017) included in the S&P500 index, a prediction of the "Adjusted Close" value for any of them, at some "future" date, is to be made. By "future" it is meant that the date must be after the dates used as training data. The necessary time period of data needed for the prediction, as well as the prediction expected accuracy in future dates, are to be explored in the project, and the solution should take that into account.

As a secondary problem, "BUY/SELL" recommendations shall be given by a Trading Recommender System that uses the trained model. The goal of the Trading Recommender System is to try to maximize profit in a certain time horizon. Given today's values of (Open,High,Low,Close,Volume,Adjusted Close), it will produce a list of trading orders (including the possibility of an empty list), that are recommended to be executed tomorrow, just before the closing time (assume 20 minutes before closing time, for example). The recommender would have been trained with historical data, and will continue to learn from new data, as it arrives.

Note: This proposal is inspired in the Assignments of the "Machine Learning for Trading" course, by Tucker Balch (Udacity) and it's corresponding course in Georgia Tech ([Assignments [3]]). The soultion is expected to be a slight improvement (or at least an interesting experiment), combining Supervised Learning and Reinforcement Learning on the same trading system.

Datasets and Inputs

The dataset to be used consists of the (Open,High,Low,Close,Volume,Adjusted Close) daily values for the stocks that are presently (as of January 17th, 2017) included in the S&P500 index, in the range dates from January 1st, 1993 until December 31st, 2016. All the features are taken daily, and in market hours (normally from 9:00 to 16:00). “High” and “Low” are the highest and lowest price values for a stock, in that day. “Open” and “Close” are the price values at the begining and ending of the market hours for a particular day. “Volume” is the value of the total trades on the stock that were executed on a day. “Adjusted Close” is the same as “Close”, but “filtering” some artificial price variations due to stock splits and dividend payments.
The S&P500 index value (taken from the SPY ETF), in the same date range, will also be added as input data. The date range that the algorithm will consider may not be the entire period, but it potentially could.
The source for the input data is Yahoo! Finance, and the data can be obtained with the “pandas_datareader.data.DataReader” function.

The dataset will be divided in a training set and a test set, considering time as the dividing factor (the test set is totally in the future of the training set). For validation, there could be a further “time division” of the training set, or a “rolling” validation could be implemented.

Solution Statement

The solution has two parts:

- First, a Supervised Learning algorithm will be trained to predict the value of the stocks in the dataset. The predictions will be produced for the next day after the training period, for the next 7 days, 14 days, 28 days, and 56 days. The predictions for the next day will be obtained by direct application of Supervised Learning algorithms (regression algorithms, most liekely). For the prediction of “further ahead days” two approaches will be considered and compared. The first one involves training many Supervised Learning Algorithms, one for each day ahead that is required to predict (the labels for the “next day predictor” would be taken from the next day, the ones for the “2 days ahead predictor” from the day after, and so on). The second approach involves using the “next day predictor” in a recursive way: to predict the “2 days ahead value” the “1 day ahead” predictor would be used first, and its predictions would serve as input for the same “1 day ahead” predictor with the “time base” shifted (that way it would be effectively predicting the value of “day 2 from now”).

- Second, a Reinforcement Learning algorithm will be implemented to make trading suggestions for the next day. In the case of the Q-Learning algorithm (some others may be implemented too) the state will be composed of some (discretized) Technical indicators (e.g.: Bollinger values, Momentum, Relative Strength Index, etc.) for each of the stock symbols, and the (discretized) amount of shares owned for each stock (which can be positive or negative). The rewards will, initially, simply be total value of the portfolio (cash included). Some “delayed reward” may be explored (cumulative returns in “n” days, Sharpe Ratio after “n” days), if there is enough time, or if the results with the “total portfolio value” are not good enough.
  The algorithm will take the predictions of the Supervised Learning algorithm as an input to estimate the Transition matrix and the Reward matrix. That way those matrices can vary in time (the RL part is not trying to converge to constant R and T matrices, but the SL algorithm is estimating those at each point in time).

Benchmark Model

The proposed benchmark for the model is the S&P500 index (SPY) cumulative return in the period in which the recommendations are followed. That is a standard benchmark for trading strategies, as the “Efficient Markets Hypothesis” would state that it is not possible to beat it. To compare the benchmark model with the one created in this project, the final Cumulative Return and the Sharpe Ratio are proposed as metrics.

Evaluation Metrics

For the prediction part, one of the proposed metrics is the RMSE of the predictions against the historical values for the periods defined (7 days, 14 days, 28 days, and 56 days after the last training sample). To get a single number (there will be about 500 stocks to test) the simple average should be taken (and its standard deviation reported, also). The other proposed metric is the average relative prediction error on the final dates of those periods (day 7, day 14, day 28, day 56) among all the predicted stock values. In particular, a goal of the project is to achieve a mean relative error of less than 5% in day 7.
The formulas for the metrics can be seen below (the index “j” refers to the stock that is being taken into account, and “i” is the “time” index).

$$\mu^j = \sum_i y_i^j$$

$$RMSE^j = \sqrt{\frac{\sum_i (ypred_i^j - y_i^j)^2}{\sum_i (y_i^j - \mu^j)^2}}$$

$$AvgRMSE = \sum_j RMSE^j$$

Mean relative error:

$$MRE^i = \sum_j \left| \frac{ypred_i^j - y_i^j}{y_i^j} \right|$$

For the automatic trader (the suggestions made by the Android App), the metrics to use will be the total cumulative return in the periods, and the Sharpe Ratio in that period. Those metrics will be used to compare against the S&P500 index as a benchmark.

$$CumRet^j = \frac{value_i^j}{value_0} - 1$$

And the Sharpe Ratio in the period is the one below (the free interest rate will be considered zero). “mean” is the regular mean, and “std” is the standard deviation.

$$SR^j = \sqrt{SamplesPerYear} \times \frac{mean(dailyReturn^j)}{std(dailyReturn^j)}$$

Project Design

i. Get the data.

   i. Get the S&P500 constituent list as of today.
   ii. Download all the values from January 1st, 1993 until December 31st, 2017.
   iii. Preprocess the data (eliminate market holidays, fill the missing values, standarize when necessary, etc)
   iv. Separate the test set.
   v. Save all the data in an easily maneagable format for the Python scripts (probably a “pickle” file)

ii. Predict next day prices

   i. Define the target values (“y”) to be the Adjusted Close value of the next day after the training period, and the features (“X”) to be all the known values (including Adjusted Close) of the “base period” (from some days before today until today).
   ii. Select the algorithms to use. Some possible choices are: Logistic Regression, Knn, Random Forests, and AdaBoost.
   iii. Find the optimal “base period” and “training period” to get the best results. That means:
      - “base period”: How many days of historical data are included in the features each time.
      - “training period”: How many days back should I look for the training data. The training period is larger than the base period, and many base periods may fit the training period. Each possible base period inside the training period, for each stock, is a sample of the training set.
   iv. Grid Search for hyperparameters. Choose optimal model and hyperparameters. (iterate with the previous point, or consider the base period as another hyperparameter)
   v. Evaluate the RMSE and Mean Relative Error with the test set. In particular show the variations of the MRE in “future” times (“future time validity of the model”).

iii. Predict future day prices

i. Build a direct model modifying the algorithm for one-day predictions, by using as the target value (“y”) “the Adjusted Close some days later”.
ii. Build a recursive method, using the “one-day predictor” and moving the training set window (using the “next day predicted value” as input data, and predicting the “2 days ahead value”).
iii. Compare the performance of both methods.
iv. Evaluate the RMSE and Relative Mean Error with the test set. In particular show the variations of the MRE in “future” times (“future time validity of the model”).

iv. Build a Trading Recommender System

i. Find some interesting Technical Indicators to use as the state for the system.
ii. Build a Market Simulator.
iii. Implement Q-Learning, Dyna and SARSA.
iv. Test the results with the proposed metrics.
v. Add the price value predictor as part of the Dyna model
vi. Test the final results

---

i. [Dai] http://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf “Machine Learning in Stock Price Trend Forecasting - Yuqing Dai, Yuning Zhang” ↩

ii. [Madge] https://www.cs.princeton.edu/sites/default/files/uploads/saahil_madge.pdf “Predicting Stock Price Direction using Support Vector Machines - Saahil Madge, Advisor: Professor Swati Bhatt” ↩

iii. [Assignments] http://quantsoftware.gatech.edu/CS7646_Fall_2016#Assignments ↩