Machine Learning Engineer Nanodegree

Capstone Proposal

Gil Akos January 4th, 2017

Proposal // Deep Learning Stock Value Predictor

Domain Background

Investment firms, hedge funds, and automated trading systems have used programming and advanced modeling to interact with and profit from the stock market since computerization of the exchanges in the 1970s[1]. Whether by means of better analysis, signal identification, or automating the frequency of trades, the goal has been to leverage technology in order create investment systems that outperform alternatives - either service providers (competitors like alternative hedge funds) or products/benchmarks (ETFs or the S&P 500).

Today, the most promising and adcendant technology, Deep Learning, is the target of incorporation into advanced investment systems[2] offered by "Artificially Intelligent Hedge Funds"[3] and "Deep Investing"[4] as-a-service startups, with claims of outperformance of the S&P 500 Index of up to 87%[4]. Given profit opportunity that large, can a basic Deep Learning model built with publicly available technology achieve positive predictive performance? Even an order of magnitude less of an advantage (8.7%) over the noise in the market and a baseline of the S&P ETF (SPY) could be valuable for an amateur investor!

This project seeks to utilize Deep Learning models, specifically Recurrant Neural Nets (RNNs), to predict stock prices. Much academic work has been developed using this technique[5], as well as similar studies using Boltzmann machines[6] for both momentum trading strategies and time series prediction. As discussed above and in the below articles from sources ranging from technology magazines (Wired[3]) to the standard bearer for market information (Financial Times[2]), these models are also being applied to real world trading platforms[7]. In this study, I will use Keras[8] to build a RNN to predict stock prices using historical closing price and trading volume and visualize both the predicted price values over time and the optimal parameters for the model.

Problem Statement

The challenge of this project is to accurately predict the future closing value of a given stock across a given period of time in the future. The hypothesized solution is that a Keras RNN model is trainable across a time series[9] of adjusted closing stock price values. Because almost all stocks have a large range of historical daily closing price values, the model should be trainable over a long period of time and the subsequent predicted models backtested for their accuracy against a sizeable period of time as well. The predicted values will be logged daily both as prices and as a measured percentage difference of predicted value over actual value. The model will be applied to ten popular stocks traded on the New York Stock Exchange (NYSE) to replicate the problem and

compare results. Time allowing, I will also add states of buy, sell, hold the virtual portfolio of the project to measure gains relative to the benchmark S&P ETF.

Datasets and Inputs

All of the necessary data for the project will come from Yahoo Finance[10] and its use will be automated by way of the Yahoo Finance API[11] for ease of reproducability. The program will be given a stock ticker symbol as target for prediction and will pull the below historical data from the API to train the model: - Target Stock // Daily Adjusted Closing Price // Value we are trying to predict - Target Stock // Daily Volume // Indicates level of trade activity and will give a sense of market momentum for the individual stock - SPY ETF // Daily Adjusted Closing Price // Indicates macro market trend for prices

Additional inputs to experiment with adding to the model: - Target Stock // n Day Rolling Mean Average from Daily Adjusted Closing Price // If n=20 this could smooth out the noise in the market relative to the target stock - SPY ETF // n Day Rolling Mean Average from Daily Adjusted Closing Price // If n=20 this could smooth out the noise in the market relative to the macro market trends - Target Stock Sector ETF // Daily Adjusted Closing Price // Indicates micro market trend for prices - Target Stock Sector ETF // n Day Rolling Mean Average from Daily Adjusted Closing Price // If n=20 this could smooth out the noise in the market relative to micro trends

Solution Statement

The solution to this project will be programmed in a Python Notebook for ease of reproducability. Using a Keras[12]implementation of the TensorFlow[13] library, the solution will utilize a Recurrant Neural Net with a Long Short-Term Memory model capable of learning from time series data and will be supported by Pandas DataFrame library for convenient time series data schema after requests received from the Yahoo Finance API[11]. Additionally the project will experiment with improving the RNN with Ensemble techniques. The measures of performance will be based on the predicted stock ticker price in comparison to both the actual price and the benchmark model's predicted price which will utilize a Linear Regression model from Scikit-Learn[14].

Benchmark Model

This project will use a Linear Regression model as its primary benchmark. This model will be based on the examples presented in Udacity's Machine Learning for Trading course[15]. Additionally I will reference two other models for comparison of a Deep Learning approach to this problem - Deep Learning for Multivariate Financial Time Series[16] for an overall error rate comparison and Machine Learning with Financial Time Series Data[7] for a technical approach. Both use Deep Learning models for Financial Time Series predictions as well as focus on stock or market based applications.

Evaluation Metrics

The measures of performance for this project will be the mean squared difference between predicted and actual values of the target stock ticker at market close and the delta between the performance of the benchmark model (Linear Regression) and our primary model (Deep Learning).

The Lucena Research tool shown in the Machine Learning for Trading[15] course displays a 3.64x improvement over the S&P benchmark from 1/1/09 to 6/18/15, which I will use for both the test date range and performance evaluation per the evaluation metrics.

Project Design

This project will be implemented through the Keras/TensorFlow library using Recurrant Neural Networks to incrementally build the complexity of a Long Short-Term Memory model to accommodate the Time Series characteristic of our stock price data. Development workflow will follow the below sequence:

0. Set Up Infrastructure

- Python Notebook
- Incorporate required Libraries (Keras, Tensorflow, Pandas, Seaborn)
- Git project organization
- Set up documentation template

1. Prepare Dataset

- Incorporate Yahoo Finance API to request stock data
- Test request and successful receipt of a target stock ticker
- Process the requested data into Pandas Dataframe
- Develop function for normalizing and un-normalizing price and volume data
- Append columns of normalized data to stock Dataframe
- Plot sequnce to confirm calculations
- Convert the above sequence into a modular function
- Dataset will be used with a 80/20 split on training and test data across all models

2. Develop Benchmark Model

- Set up basic Linear Regression model with Scikit-Learn
- Calibrate parameters
- Log parameters for Step 3 and reproducability
- Log results

3. Develop Basic Deep Learning Model

- Set up basic RNN model with Keras utilizing parameters from Benchmark Model
- Calibrate parameters
- Log parameters
- Log Results

4. Improve Deep Learning Model

- Develop, document, and compare results using additional labels for the Deep Learning Model (see Datasets and Inputs section above)

- Develop, document, and compare results using Ensemble techniques
- Log Parameters
- Log Results

5. Document and Visualize Results

- Plot Actual, Benchmark Predicted Values, and Deep Learning Predicted Values per time series
- Plot Benchmark performance versus Deep Learning performance as described as "delta" above
- Plot Seaborn Grid of additional label use and corresponding improvement
- Analyze and describe results for report.

Footnotes

1. Bloomberg // History of Algorithmic Trading Shows Promise and Perils
2. Financial Times // Money managers seek AI's 'deep learning'
3. Wired // The Rise of the Artificially Intelligent Hedge Fund
4. Stocks Neural Net // About
5. Deep Learning for Time Series Modeling
6. Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks
7. MIT Technology Review // Will AI-Powered Hedge Funds Outsmart the Market?
8. Keras // Deep Learning library for Theano and TensorFlow
9. Time Series Prediction With Deep Learning in Keras
10. Yahoo Finance
11. Yahoo Finance // Python API
12. Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
13. TensorFlow Tutorial for Time Series Prediction
14. Scikit-Learn
15. Machine Learning for Trading
16. Deep Learning for Multivariate Financial Time Series
17. Machine Learning with Financial Time Series Data