Capstone Project Rohit Gupta

Machine Learning Engineer Nanodegree 12 June 2018

Stock Price predictor with Recurrent neural nets

PROJECT OVERVIEW

The transition from traditional auction in 1970s to computerized transactions was fuelled with a need for an efficient access to the market. This set a stage for algorithmic and high-frequency trading. An estimated 70% of US equities in 2013, accounted for by automated trading [1]. Investment firms and hedge funds have used mathematical modelling to predict market trends. They aim towards maximizing the return and spreading their risk over different financial assets. These approaches can be classified as *Follow-the-Winner*, *Follow-the-Loose*', *Pattern-Matching* and *Meta-learning*. Though these traditional methods are efficient, their performance is dependent on the validity in different types of markets.

Quantitative strategies of hedge funds have received considerable returns over decades. Application of growing computing power and availability of big data has allowed models to identify and harvest on market inefficiencies [2]. These organizations are actively looking for AI solutions to outperform the benchmark indexes. In recent times a large ecosystem of start-ups has unfolded, centred around FinTech, with the aim of providing AI solutions for investment and saving.

The project aims towards utilizing Recurrent Neural Nets to predict the stock prices. The goal here is to leverage modelling abilities of neural networks for time series forecasting [3]. Time series forecasting is a difficult type of predictive modelling problem, as it has added complexity of sequence dependence among the input variables [7]. Recurrent neural networks have been quite successful in modelling time series data. LSTM is a type of RNN which can model short-term memory over a long period of time. LSTM enables us to model sequence dependence as short-term memory and incorporate it as a feature in our model.

PROBLEM STATEMENT

The goal of this project is to predict future adjusted price for a given stock across a given period. With vast amount of historical data, application of machine learning techniques

becomes appropriate here, to predict price trends for a stock being traded in a market. These models should be tested for accuracy and validated for performance. Because of the availability of wide range of historical data, a model of this type can be trained over a long period of time and can make predictions with a time series.

This project aims to answer these questions

- Can a deep learning model beat a machine learning benchmark model for stock price prediction?
- Which architecture/ hyperparameters is gives an optimal result for the model?
- Can a simple deep learning model perform good enough to drive the investment strategies?

METRICS

It is important to measure the quality of a model by quantifying its performance. Here we are going to use Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). It is the difference between the price predicted by a model and actual price, a stock was trading on a day. Mathematical formula of RMSE is as depicted in the Fig 1. Using RMSE gives us consistency while comparing with other studies. Also, we will use graphical visualizations of different models to find degree of fit, ability to incorporate volatility and lag across time and other intrinsic micro trends from dataset.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Fig 1. Root Mean Square Error (RMSE)

ANALYSIS

DATA EXPLORATION & EXPLORATORY VISUALIZATION

In this project we are going to analyse stocks of Apple (AAPL), Google (GOOG), Amazon (AMZN) and SPY stock indices. The historical data for these entities was retrieved from Yahoo finance application and is available in repository inside 'Project/data' folder. To keep the report brief and interesting, in this report we are going to discuss only Apple stocks. Although, a separate notebook for each entity having detailed analysis can be found inside the 'Project' folder. All the images of the report can be found in 'Project/images' folder.

Here, we are using pandas to load a csv file for a given stock (in our case ‘data/AAPL.csv’). Date column is parsed while loading and set as an index. As we are going to use ‘Rolling mean and Adjusted closing price as a feature for our model, all the columns except this were

dropped. For rolling mean, window size of 10 is selected. It is evident from the Fig 2 that resolution of window size 10 is better than the other window size.

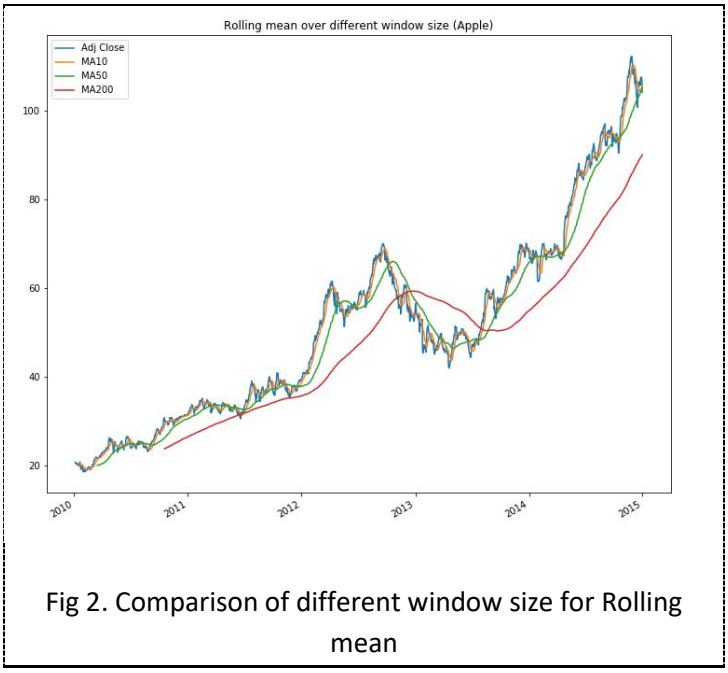

Fig 2. Comparison of different window size for Rolling mean

We are using past five-year data to create and validate models. Data ranges from Jan 2010 to Jan 2015. Fig. 3 depicts Adjusted closing price for stocks in these date range. For Adjusted closing price it looks like google has always been much more valuable as a company. But to better understand this we must look at total market cap of the company not just the stock prices. This information though absent, can be derived from total value traded for a given period. In Fig 3, we can also see volume trade plots for different stock, using both values, market cap of a company can be approximated.

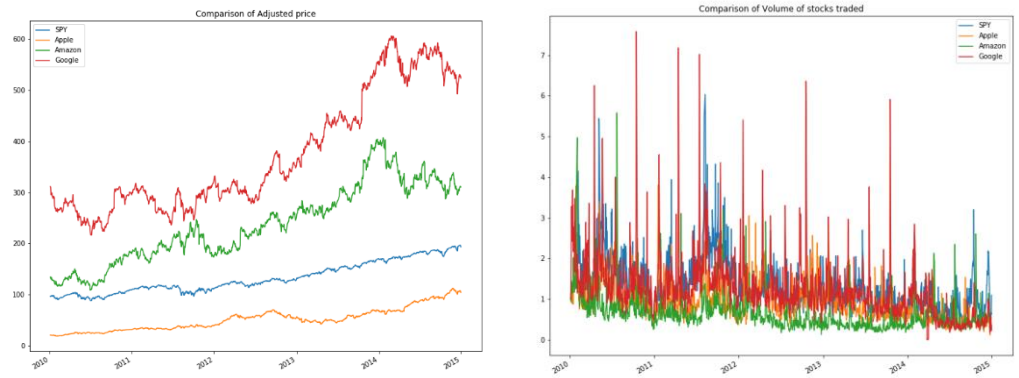

Fig 3. Adjusted closing price for Apple, Google, Amazon and SPY(left) Volume of the stocks traded (right).

These entities share a common domain, i.e. software industry. This correlation can be

explored. Correlated entities react in a similar fashion to external events to the industry they belong. For example, local government policies, effect of season and climates etc. In Fig4, we can observe that these is a certain degree of correlation between these stocks.

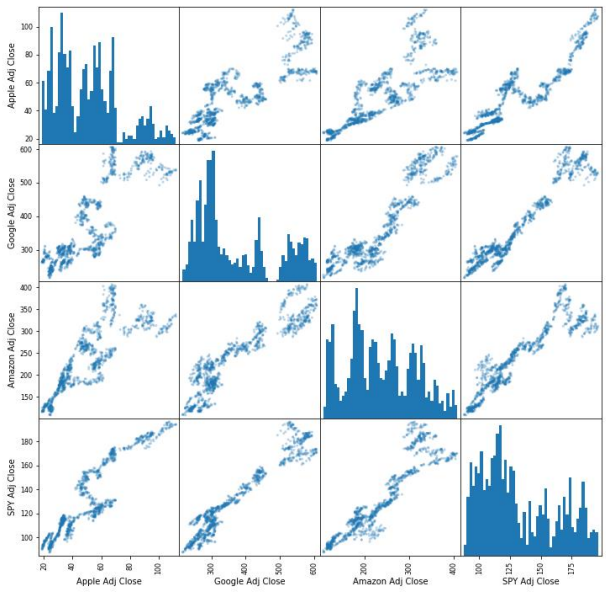

Fig 4. Correlation matrix for stocks (Apple, Amazon, Google, SPY)

Daily percentage return is a good metric to estimate daily volatility of a given stock. It basically indicates, if a stock was bought today and sold tomorrow, how much gain or loss it will make. Fig5 plots the daily percentage returns of stocks. It can be observed, all the stocks we are working with, has a similar degree of volatility.

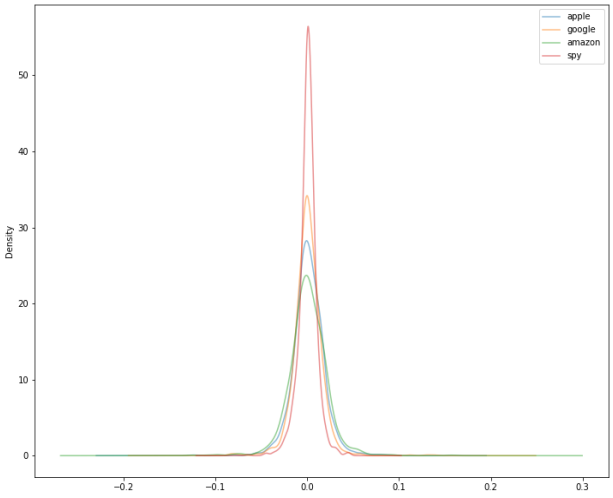

Fig 5. Daily percentage return for stocks (Apple, Amazon, Google, SPY)

While daily returns are useful, it doesn't give the investors an immediate insight to the gains they have made till date, especially in cases when stocks are very volatile. Cumulative returns are computed relative to day investment was made. If cumulative returns are above 1, it is making profit and vice versa. Let's explore cumulative returns. In Fig6, it can be observed that Apple gave a good cumulative return compared to others. In Fig7, correlations between returns of different stocks can be observed.

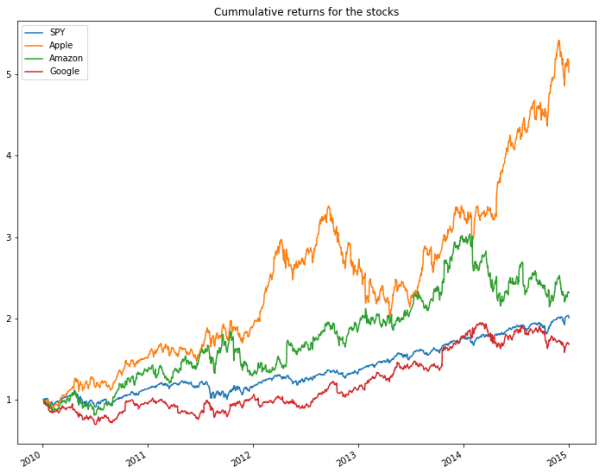

Fig 6. Cumulative return for stocks (Apple, Amazon, Google, SPY)

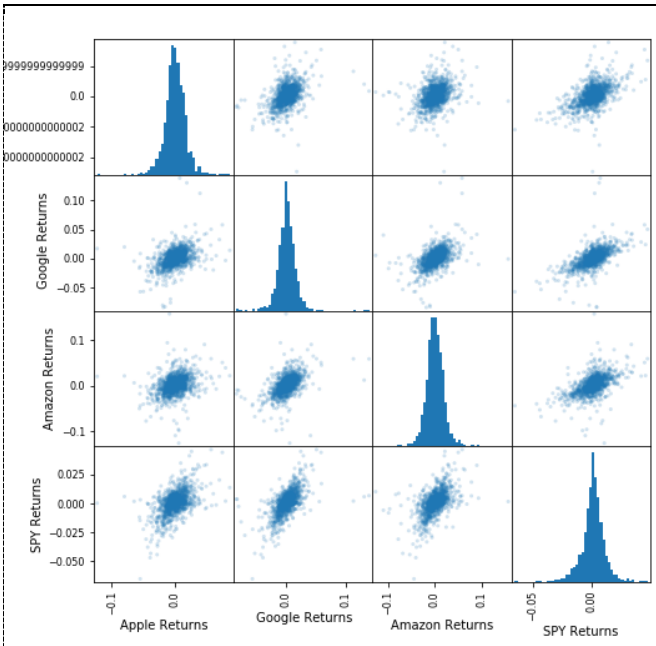

Fig 7. Return correlation for stocks (Apple, Amazon, Google, SPY)

ALGORITHMS AND TECHNIQUE

Algorithm

The problem statement of this project revolves around making predictions for a given time series of stock prices. The project includes studies utilizing Basic RNN, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) cells. Standard RNNs suffers from vanishing and exploding gradient problems. Because of which, learning long term temporal dependencies are difficult in them. LSTMs and GRUs are types of RNN. They use special units (and gates) in addition to standard units. A set of gates in them, allow to control when an information is remembered, when it is forgotten and when it is passed to output. This architecture allows them to learn long term dependencies. GRUs are like LSTMs but uses simplified structures. They use gates to control the flow of information but uses fewer gates and no separate memory cells.

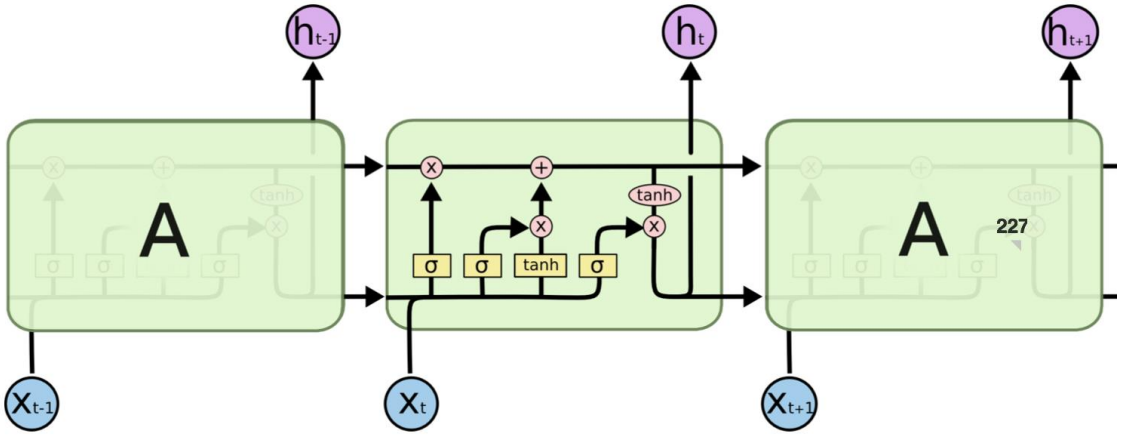

Fig 7. Return **FIGURE OF A LSTM**

The project includes experiments with these three types of RNNs, with a goal for increasing the prediction accuracies of stock price.

Algorithm Description and Justification

A normal feed forward networks can learn a relationship between inputs and outputs. They have limitations when ordering of data matters, and output at a given time step depends on the events happened in previous time steps, e.g. stock prices. In such cases, a network needs to incorporate information from previous event to give a sensible prediction. RNNs outshines here as it maintains a hidden state which is passed from previous time step, back into the network at the next time step. Hidden recurrence learns what to remember, what to forget and what to pass to the network. In this case input to the

network is hidden state from previous time step along with the input. The flow of this values is controlled through different gates, parameters of which are learned during training.

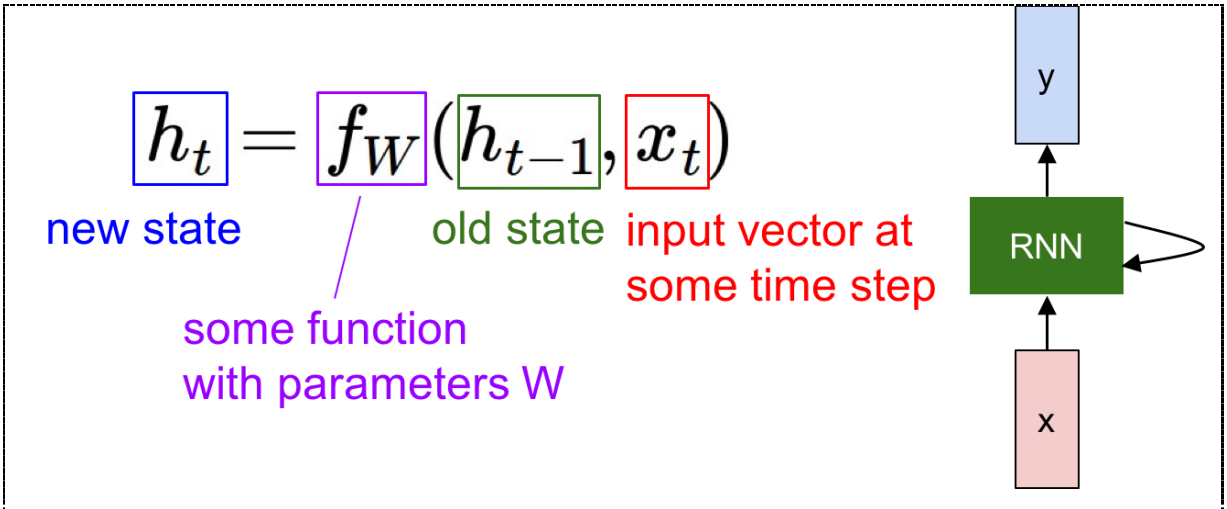

Fig 7. Return **FORMULA diagram https://www.yuthon.com/2016/10/30/Notes-for-CS231n-RNN/**

From above formula it can be observed that current hidden state h(t) is a function of previous hidden state h(t-1) and the current input x(t). The network learns to use h(t) as a lossy summary of the task-relevant aspects of the past events. Total loss for a given window/sequence of inputs paired with sequence of targets will be the sum over all time steps.

The RNNs are promising with respect to learning temporal dependencies in input data. The available context in hidden states may allow RNNs to learn trend (upwards, downwards) and seasonality (repeating patterns over time) in data. Traditional time series methods use univariate data with linear relationships and fixed and manually-diagnosed temporal dependence. RNNs add the explicit handling of ordered observations and the promise of learning temporal dependence from hidden state context. This makes RNNs suitable for the problem statement of this project **[https://machinelearningmastery.com/promise-recurrent-neural-networks-time-series-forecasting/]**.

Algorithm Parameters

In addition to optimize model architecture, following set of parameters can be tuned to improve the accuracy and reliability of the model predictions. In project, these parameters are encapsulated in a class, instance of which is used while training and testing. It can be found at 'Project/deepstocks/common_configs.py'.

- Input parameters
    - Pre-processed and normalized Adjusted closing price
    - Rolling mean (window=10) of Adjusted closing price

    - Keep probability
- Neural network architecture
    - Model type (LSTM, GRU, BasicLSTM)
    - Number of nodes or neurons per layer (tested 64, 100, 128, 156, 256, used: 156)
    - Number of layers (tested 1, 2, 3, used:2)
- Training parameters
    - Train/validation/test split (proportion of data for training, validation and testing, used 70%, 15 %, 15% respectively)
    - Batch size (how many time steps to include during single time step, tested: 50, 64, 128, used: 64)
    - Optimizer function (function to optimize and minimize the loss, used Adam optimizer throughout)
    - Epochs (how many times to train over the dataset, tested: 28, 56, 64, 150, 200, used: 64)

Techniques

1. Loading the data
    - A dataset of a given stock is loaded into a pandas dataframe for a given date range.
    - Date values is set as index, and all the columns except 'Adj Close' and 'Volume' are dropped from dataframe.
    - 'Adj Close' column is normalized to have values between 0 and 1.
    - A column with name 'Rolling mean' is added to the dataframe, the values of which is set to rolling mean on 'Adj Close' values with a window of 10 (using pandas.Series.rolling).
2. Train-valid-test split
    - For training the model, 'Adj Close' and 'Volume' features from the above dataframe is used.
    - This sequence is split into 70:15:15 ratio, 70% for training and other two for validation and testing.
3. Tensorflow graph
    - A graph of model is created using different tensorflow operations.
    - Different summaries are written to logs, to visualize better with tensorboard.

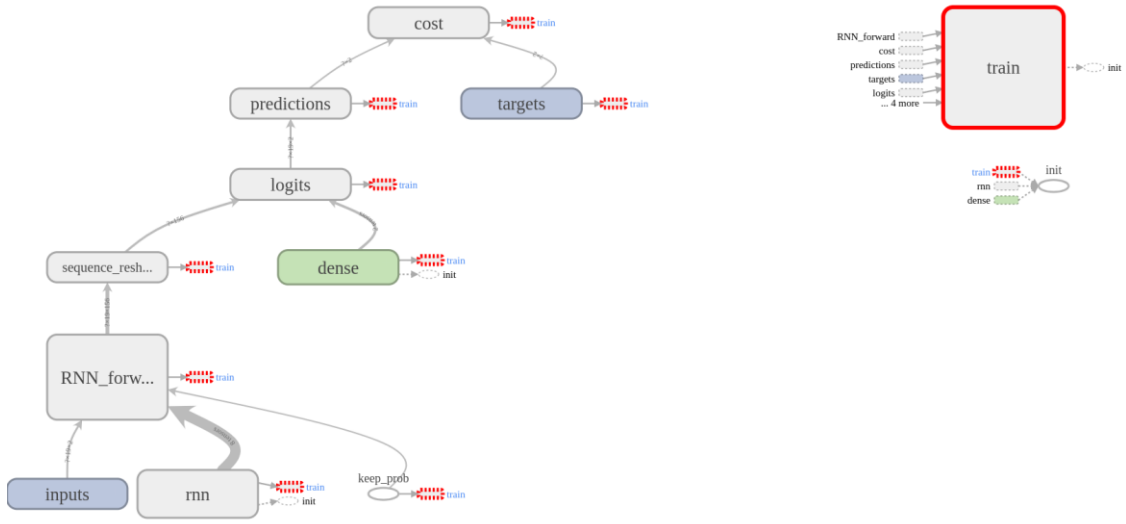

Fig 7. Return correlation for stocks (Apple, Amazon, Google, SPY)

BENCHMARK

The benchmark for the given project is a linear regression model. The aim here is to build a model having better performance than this benchmark.

METHODOLOGY

DATA PREPROCESSING

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Duis autem

IMPLEMENTATION

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel

REFINEMENT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

RESULTS

MODEL EVALUATION AND VALIDATION

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Duis autem

JUSTIFICATION

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel

CONCLUSION

FREE FORM VISUALIZATION

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Duis autem

IMPROVEMENTS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel