Machine Learning Nanodegree Capstone Proposal

Domain Background

The quest for predicting the stock prices has began since the inception of the stock market. Different approaches have been developed including the fundamental analysis, the technical analysis and the current hype, artificial neural network based approach[1].
Some people find it one of the most intriguing and intellectually challenging tasks in the intersection of finance and machine learning. Others are simply motivated by the money, referring to predicting stock price as “digging gold” or “printing money”. For me, it’s a bit of both.
It’s been recognized that the stock market is a fairly complex system and stock price is influenced by a large number of factors. That is to say, a variety of input sources could be leveraged for stock prediction. Particularly, some researchers have attempted to develop algorithms that gain insight on the short term trajectory of certain stocks by digesting the media (textual analysis)[2]. Some have used a more holistic approach taking into account the history of the stock price, the financial news, the financial statements etc.[3]
I was able to find two academic papers that report using RNN for stock price prediction and bitcoin price prediction respectively.[4][5]

Problem Statement

For this project, I would like to scope the problem to “building a stock price predictor that takes daily trading data over a certain date range as input, and outputs projected estimates for given query dates. Note that the inputs will contain multiple metrics, such as opening price (Open), highest price the stock traded at (High), how many stocks were traded (Volume) and closing price adjusted for stock splits and dividends (Adjusted Close); your system only needs to predict the Adjusted Close price.” With this definition, the data source is confined to only stock price historical data. From a machine learning perspective, this is a regression problem.

Datasets and Inputs

There are many open source stock price data that is readily available from vendors like Yahoo! Finance, Bloomberg API, Quandl, Kaggle etc. For this project, I’ll be primarily using a dataset from Kaggle that contains the stock price of google from 5/22/09 to 5/3/17. The features of the dataset include “Open”, “High”, “Low”, “Close”, “Volume” and “Adj Close”. Particularly, the “Adj Close” will be the prediction target. The most recent 10% of the data will be used as testing data, and the rest of the data will be used in training and evaluation(8:1:1 for training:evaluation:testing).

Solution Statement

The proposed solution is a python script that builds, trains and tests a model(using RNN) that predicts stock price. The model will be trained with historical stock price data.

Benchmark Model

I propose a supervised learning model that uses SVM as the benchmark model. Specifically, the benchmark model should be less complex than the model proposed in the solution but renders a certain amount of accuracy compared to random guessing. The benefit of using SVM is that it allows a more flexible relationship between the input and the output through choosing different kernels.

Evaluation Metrics

The primary evaluation metrics that will be used is MSE(mean squared error) or RMSE(root mean squared error).
The formula for this metrics is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE measures the difference between values predicted by the model and the values actually observed. It is considered to be one of the most popular metrics for evaluating regression models.[6]

Project Design

This project would follow a typical machine learning process, which breaks down to:

Data preparation and general cleansing

In this step, the data would be loaded and a general data munging would be conducted to prepare the data for further analysis and training the model. The format of the data would also be adjusted if necessary to facilitate analysis. According to this tutorial[7] on stock prediction by RNN, one challenge that the author had to solve was that the model needs to make predictions that are much larger than the training data. The solution the author proposed was data normalization in each sliding window, which follows the formula below:

$$W'_t = (\frac{p_{tw}}{p_{tw-1}}, \frac{p_{tw+1}}{p_{tw-1}}, \dots, \frac{p_{(t+1)w-1}}{p_{tw-1}})$$

Exploratory Data Analysis

In this step, exploratory analysis is performed to understand data better, including the type of data, relationships, trends in the data. In particular, visualizing different features plus some other statistical variables like the rolling average along the timeline provides a basic understanding for how the different factors changes and how they're interrelated with each other and the adjusted close price.

Establishing the Benchmark

In this step, the benchmark model (SVM based supervised learning) is to be built, trained and tested. One thing worth noting is that through this step, we'll gain a better idea whether feature selection/dimension reduction helps the performance.

Feature Selection/Dimension Reduction

In this step, the most relevant features are selected and some irrelevant or redundant info is removed from the dataset. Some techniques could be applied like PCA.

Model Building

In this step, a deep learning model is built. There are a few design decisions to make, specifically, 1) how many layers in the model and the number of nodes for each layer(the structure of the neural network), 2) what activation to use for each layer 3) the loss function 4) the optimizer etc.

Training and Evaluation

In this step, we need to prepare training and testing data(splitting the data into training and testing). One thing to note here is that since we always want to predict the future, we take the latest 10% of data as the test data.

Hyperparameter tuning

The purpose of fine tuning the hyperparameters is for improving the model's performance. The basic algorithm used for hyperparameter optimization is grid search. The hyperparameters that could be tuned include batch size, training epochs, optimization algorithms, learning rate and momentum, network weight initialization, activation functions, dropout regularization, the number of neurons in the hidden layer etc.

References

[1] https://en.wikipedia.org/wiki/Stock_market_prediction
[2] https://www.technologyreview.com/s/419341/ai-that-picks-stocks-better-than-the-pros/
[3] https://web.stanford.edu/class/cs221/2017/restricted/p-final/davdnich/final.pdf
[4] https://mospace.umsystem.edu/xmlui/bitstream/handle/10355/56058/research.pdf
[5] http://trap.ncirl.ie/2496/1/seanmcnally.pdf
[6] https://en.wikipedia.org/wiki/Root-mean-square_deviation
[7] https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html