Neural Networks and Ensemble Learning in Stock Price Prediction

Fong Wei Jie, Wu Yayi, Wang Yibo, Zhang Zequn

Introduction

A Neural Network is a model that operates similarly to the human brain. A neural network is made up of numerous small units called neurons; these neurons are gathered into several layers. Layers are segments of neurons that are connected to one another through their neurons. Each neuron is connected to another layers' neuron through weighted connections. The weighted connections are adjusted with a real-valued number that is associated with them. A neuron takes the value of a connected neuron and multiplies it with their connections' weight. The aggregation of all the connected neurons is the neurons' bias value. The bias value is then passed to an activation function which transforms the value and assigned it to the connected neuron in the next layer. This value is circulated through the entire network. In order to adjust the weights to fit the output better, a feedback mechanism known as backpropagation is implemented. One of the models this paper would focus on Long Short-Term Memory (LSTM), a recurrent neural network, which resolves the vanishing gradient problem during backpropagation.

Sometimes relying on the outcomes of just one model may not be enough. Ensemble learning provides a systematic solution for combining various learners' predictive power. The result is a single model that provides multiple models of aggregated performance. Bagging and boosting are two commonly used ensemble learners in decision trees. Trees are constructed sequentially in boosting, so that each subsequent tree seeks to decrease the past tree's errors. Each tree learns and updates the remaining errors from its predecessors. The tree growing next in the sequence will therefore learn from an updated version of the residuals. In this paper, we would also focus on XGBoost, a gradient boosted decision tree.

Methodology

A. Data Selection

BlackRock Energy & Resources Trust (BGR) was selected as the stock to be used for analysis. Open, high, low, close and volume data from 2005 to 2017 was obtained from Kaggle. Using an 80-20 train-test split, the test set spans a duration of 2.5 years, from mid-2015 to 2017. A plot of the closing price is shown below.

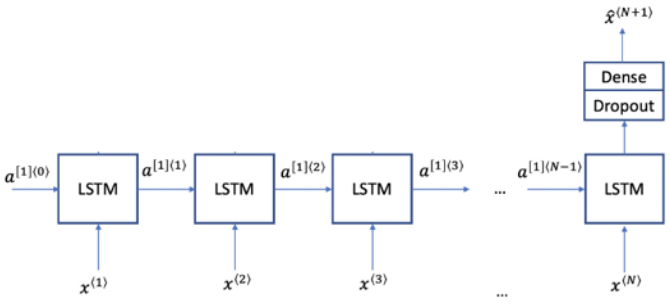
B. Data Pre-processing

We aim to predict the next day's closing prices using data from the past N days, which means a forecast horizon of 1 day. One of the hyperparameters to be tuned is N, which we set to be a range from 4 to 10 days. Train and test data were normalised using data from the past N days.

C. Trade Execution

A simple strategy was proposed; if the predicted price of the next day is larger than that of the previous day, we would either hold our position if we are already holding onto the stock, or buy 1 unit of the stock if we are not currently holding onto any. If the predicted price is lower than that of the previous day, we would sell off the stock (if any).

D. LSTM Architecture



The above shows the LSTM architecture which we used. We used one layer of LSTM modules, and a dropout layer to avoid over-fitting, and finally a fully connected dense layer before the output. A linear activation function was used; mean absolute error (MAE) was utilised as the loss; and the Adam optimiser was used.

The hyperparameters tuned included N (previous N days used to predict the stock price of the next day), the dropout rate, number of neurons in the single layer, epochs and batch size.

Results

A. Metrics

1. Sharpe Ratio: We define it to be the mean of the daily returns divided by the sampled standard deviation of the daily returns, multiplied by the square root of 252, the number of trading days per year in the USA markets. A value above 1 is considered to be acceptable.

$$Sharpe = \frac{mean(daily\ returns)}{sd(daily\ returns)}\sqrt{252}$$

Comparing the price of the next day and the current day, 2 labels were generated, *Increase* and *Decrease*. We then create a confusion matrix, which reports the number of false positives (FP), false negatives (FN), true positives (TP) and true negatives (TN). 2 more metrics were then generated from the confusion matrix:

2. Precision:

$$Precision = \frac{TP}{TP + FP}$$

3. F_1 score:

We first define recall:

$$Recall = \frac{TP}{TP + FN}$$

Then F_1, which ranges from 0 to 1:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

B. Evaluation

We used a Linear Regression model as the base model, and tuned the hyperparameter, N days. N = 6 gave the largest Sharpe Ratio, which we used as the final model.

For XGBoost, we tuned the following hyperparameters: N days, number of estimators, maximum depth of a tree, learning rate, minimum sum of instance weight (hessian) needed in a child, subsample ratio of the training instances, subsampling of columns by tree and by level, and gamma.

The hyperparameters tuned for LSTM were covered in the previous section.

After hyperparameter tuning, the evaluation of the results is shown in the table below:

	Sharpe	Precision	F_1
Linear Regression	0.56	0.47	0.47
XGBoost	1.47	0.51	0.51
LSTM	1.64	0.51	0.51

Both XGBoost and LSTM performed much better than the base Linear Regression model in terms of Sharpe Ratio. The precision and F_1 metrics are also higher than the base model.

Discussion

Although it is promising to use neural networks and boosted trees as a predictive instrument, we have experienced several constraints and limitations.

The metrics precision and F_1 could have been better handled by defining a separate problem. A 2-class (*Increase* or *Decrease*) prediction problem could have been used instead of regression to predict Closing price. By changing the labels and the loss function to fit the metrics, the precision and F_1 scores would likely be much higher. This would also be in line with our trading strategy as well.

In this paper, we focused more on the optimisation of the model, over the trade execution. To improve the Sharpe ratio of all the models, other trading strategies that incorporate the magnitude of increase of the predicted closing prices with the amount of stocks bought or sold can be considered as well.

It is still possible to further improve the performance of the LSTM model in terms of the Sharpe ratio. The optimiser (set as Adam), number of layers (set as 1), regularisation method (set as Dropout), and the loss function (set as mean absolute error (MAE)) are also possible parameters to be adjusted. We tried creating a 2-layer LSTM architecture with a Dropout layer in between, but that did not perform as well as the single layer LSTM model. Perhaps more layers are required to improve the performance of the model. However, using a more complicated model would result in a longer training duration, as well as a higher chance of overfitting.

One issue with neural networks is that it has the tendency to overfit to the training data. This would make the model perform worse on the test set, or on new sources of data. One method to check for overfitting would be to use a validation set along with the training set, and the losses of both sets would be compared; if the training loss and validation loss are close to each other, then likely overfitting did not occur. To resolve overfitting, training could be stopped prematurely, or a stronger regularisation could be added.

Only one stock was randomly selected for analysis in this paper. Experiments could have been run over other stocks to compare their performances, to recognise if similar Sharpe ratio could be achieved across various stocks. The hyperparameters of the best model for each stock could also be compared to understand the main parameters to be tuned, so that the model could be easily scaled for future usage. A portfolio of stocks, or other asset classes could be considered as well.

References

Kaggle.com. (2019). *LSTM model of StockData | Kaggle*. [online] Available at: https://www.kaggle.com/johanvandenheuvel/lstm-model-of-stockdata/data [Accessed 9 Jun. 2019].

Towards Data Science. (2019). Machine Learning Techniques applied to Stock Price Prediction. [online] Available at: https://towardsdatascience.com/machine-learning-techniques-applied-to-stock-price-prediction-6c1994da8001 [Accessed 9 Jun. 2019].

Analytics Vidhya. (2019). *Predicting the Stock Market Using Machine Learning and Deep Learning*. [online] Available at: https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/ [Accessed 9 Jun. 2019].