Implementation of Time-Delay Recursive Neural Network for Classification of Security Price Jumps

Brandon Peh, Ren Hao Tan

Stock price manipulations by "pump-and-dump" scammers involve systematic trading of stocks with uncertain fundamentals and low daily trading volumes. A recurrent neural network (RNN) trained on a set of time-delayed covariates relating to technical dynamics is shown to be capable of forecasting the occurrence of these activities in the US stock markets.

Index Terms—deep learning, stock price prediction, investment science, recurrent neural network, time-series analysis (*G11, G12, G14, G17)*

I. INTRODUCTION

Significant and sudden increases in prices of stocks happen frequently due to stock market manipulations by "pump-and-dump" scammers. A pump-and-dump scheme is the illegal act of an investor or group of investors promoting a stock they hold and selling once the stock price has risen following the surge in interest as a result of the endorsement. In general, these scheme go through the below phases:

- *Phase 1:* Stocks which have (i) small market capitalization, (ii) low trading volume and (iii) volatile fundamentals are selected by scammers to be "pumped and dumped" as their prices can be more easily manipulated
- *Phase 2:* Scammers incrementally acquire long positions in these targeted stocks in a manner so as to avoid detection, i.e. no significant price movements
- *Phase 3:* Scammers trigger sudden price increases through small but expensive buy orders, alongside an aggressive stock promotion campaign; "pump"
- *Phase 4:* Scammers take profit by selling their long positions once stock price has gone up substantially; "dump"

We wish to use deep learning to identify stocks which are undergoing Phases 1 - 3 in the universe of US NYSE, AMEX and NASDAQ listed securities. This paper hypothesizes that it is possible to adequately describe Phases 1 - 3 using non-linear interactions of four time-varying covariates: prices, trading volumes, number of outstanding shares and earnings.

Intuitively, in Phase 1, many metrics which scammers could use to screen for potential pump-and-dump stock targets are derivatives of these four covariates. Example:

Pump-and-dump screening metrics	***Base fundamental metric***
Market capitalization	Number of outstanding shares, price
Price-to-earning ratio	Price, earnings

Earnings per share	Number of outstanding shares, earnings
Trading volume	Trading volume

Also, in Phase 2 - 3, the time-series variation of price and volume across time could potentially reveal the actions of scammers. Example:

Scammer actions	***Price & volume variations***
Scammers incrementally acquire long positions in these targeted stocks in a manner so as to avoid detection, i.e. no significant price movements	Consistent and significant increases in volume across a few days with minimal price impact
Scammers trigger sudden price increases through small but expensive buy orders, alongside an aggressive stock promotion campaign; "pump"	Surge in prices despite limited volume increase

II. MODEL SPECIFICATION

We assume that variations in the patterns in price-volume movements which relate to identification of "pump-and-dump" schemes are generally shift-invariant, in that they are generally independent of prior determination of the start of the pattern. Accordingly, we are able to expand the set of raw time-series covariates of a stock, i.e. daily closing price P(t), daily trading volume V(t), number of outstanding shares N(t) and trailing-twelve months earnings E(t) into multiple tuples of observation.

In particular, we assign a time delay of 20 trading days such that the covariates at any time t becomes:

$$X_t^{\rightarrow} =< P(i), V(i), E(t), N(t) \quad \forall\ i \in [t-19,\ t] >$$

This paper defines a sudden increase as a price jump of more than 50% across a five-day trading period. Therefore, the dependent variable at any time t is:

$$y_t = I(\frac{P(t+5)}{P(t)} > 1.5)$$

We train the model on available data from 1st February 2008 to 31st January 2015. Testing is then applied using data from 1st February 2016 to 31st January 2018. To reduce noise in the neural network, we reduced the set of all NYSE, AMEX and NASDAQ stocks to only those with current price < 10 and current beta > 1.0. This is because we assume that most stocks targeted for "pump-and-dump" schemes exhibit these financial attributes.

For our baseline results, we used a logistic regression model to determine the baseline results for comparison. A table of results is displayed under the results section.

The logistic regression revealed an underlying challenge in our project. On the one hand, the skewness of our data towards a non-positive response meant that there was a limited amount of learning for positive predictions. On the other hand, we are more interested in the model's ability to predict positive outcomes (which are financially profitable results).

In fact, traditional measures such as the ROC curve would mask much of the weakness in the set-up.

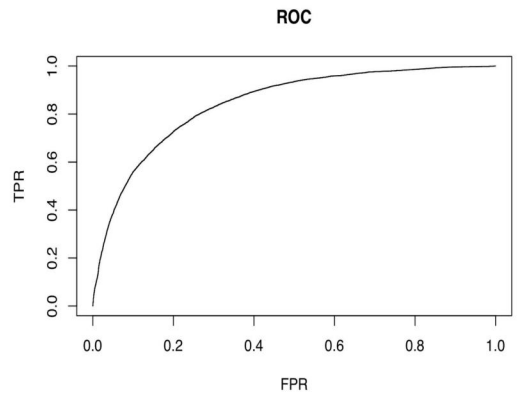

Figure 1: ROC for Baseline Logistic Model

We saw that it is theoretically possible to fine-tune the hyperparameters in order to improve precision. In this instance, we varied the cut-off for the estimator in predicting a success case.

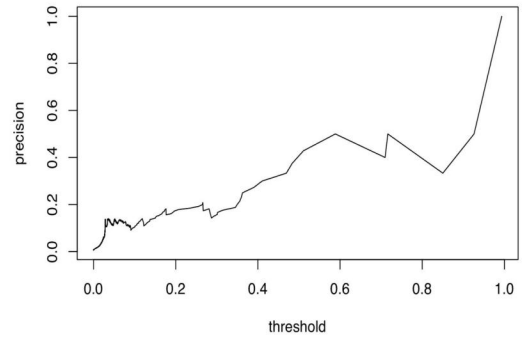

Figure 2: Precision against Threshold Variation

A high precision comes at the cost of a low TPR. In other words, we are almost dropping all the instances of stock market manipulation. It could very well be but a lucky overfitting that we can catching that one or two particular instances of TP to give a misleadingly high precision rate.

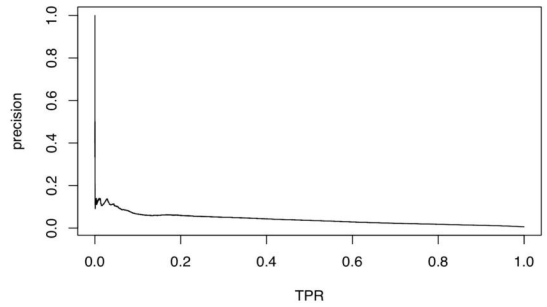

Figure 3: Precision against TPR

Despite discovering the weakness of our project set-up, we decided to go ahead and implement the TD-RNN to verify if the results could be improved. We used a four-layer RNN to improve on the baseline results. The RNN model that we used stores the variables of the preceding layer as context nodes. Past units (either inputs or hidden variables) are able to interact with subsequent units despite already having been fed-forward. The purpose of implementing such a neural network is to allow base variables like price and volume to interact non-linearly with derivative factors such price changes or price-earning ratios to codify more complex technical variations. Figure 4 illustrates how our RNN is structured.

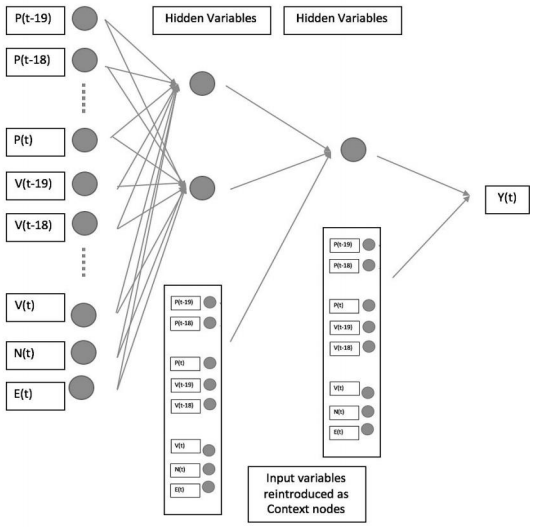

Figure 4: Structure of TD-RNN

For the hidden layers, we employ ReLu activation function to accelerate learning and avoid vanishing gradient. Since we are solving a binary classification problem, a Softmax classifier is used for the outcome layer to generate an predictor between 0 and 1. Mathematically, the RNN is described as follows:

$$h_t^{[1]} = ReLu(W_h^{[1]} X_t + b_h^{[1]})$$

$$h_t^{[2]} = ReLu(W_h^{[2]} h_t^{[1]} + U_h^{[2]} X_t + b_h^{[2]})$$

$$\tilde{y}_t = Softmax(W_y h_t^{[2]} + U_y X_t + b_y)$$

$$\hat{y}_t = I(\tilde{y}_t > 0.5)$$

where $h_t^{[i]}$ represents the i-th hidden layer and *W, U* and *b* are parameter matrices.

For the initialization of these matrices (i.e. W, U and *b*), we used the Xavier initialization as discussed in lecture to ensure that the gradients can be propagated through the neural network without vanishing or exploding.

An investor who uses our model in an attempt to profit from anticipated price increases would (i) incur additional risk from each positive prediction and (ii) profit only from each true positive prediction. In other words, he is primarily concerned with the maximization of precision (i.e. true-positive to positive ratio). Therefore, we use a weighted cross entropy loss function shown below to maximize our precision.

$$\mathcal{L} = -\sum \beta y_t \log(\hat{y}_t) + (1 - y_t) \log(1 - \hat{y}_t)$$

III. RESULTS

The following table is a summary of the results that we obtained from the different models and loss functions that we have implemented.

Model	Loss Function	Training Precision	Test Precision
Logistic Regression	Cross Entropy	5.069%	4.000%
Logistic Regression	Weighted Cross Entropy	6.711%	5.769%
TD-RNN	Weighted Cross Entropy	See Figure 2	See Figure 3

Figure 5:Training precision of TD-RNN against the number of epochs trained

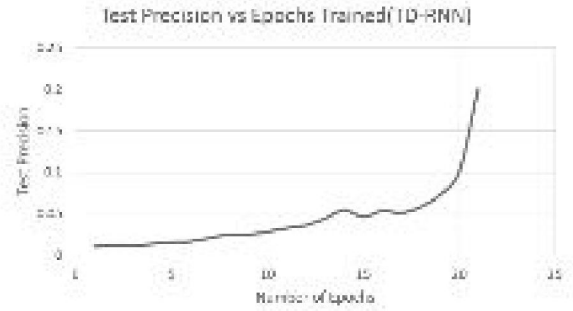

Figure 6:Test precision of TD-RNN against the number of epochs trained

IV. DISCUSSION

The results obtained were not surprising. Given a weighted cross entropy loss function, we can make the cost of predicting a false positive higher, thus improving our precision greatly. This is a similar exercise to fine-tuning the threshold level for a logistic regression as demonstrated earlier. The allowance of non-linear variations in a TD-RNN allows for some improvement; however, this improvement in precision is not sufficient for the model to be applied to a real security with huge price movements. This is because the model may be able to predict a true positive accurately, but it gives too many false negatives. Therefore, most of the time, the model is unable to identify other true positives that may occur and this is an area that requires further research.

Another weakness in this project could be attributed to the way we split the train-validate-test sets–we did so after the application of a shifting time-delayed window, not before. As a result, our test set did not vary substantially from the training set; this could be a source of overfitting. In future, it could be better to differentiate the examples temporally.

V. FUTURE WORK

The nature of the project–which is to identify rare and arguably idiosyncratic occurrences in a noisy dataset meant that much of the weaknesses demonstrated could not be ameliorated entirely. We hypothesize that it may however be possible to reduce the skewness of the dataset by changing the problem into one which asks "given that we know stock X is a pump-and-dump stock, could we then predict when the scam would have occured?". Such a project specification would drive out any idiosyncratic effect while greatly reducing the skewness of data.

In future, we would think it is possible to complement the classification TD-RNN with a LSTM neural net model which describe the price movements for each security. An investment decision to purchase a security can then be based on an ensemble learning combination of both models - one indicating a likelihood of a price jump and another forecasting future prices.

VI. CONTRIBUTIONS

Brandon: Wrote all of the code for collecting the data, preprocessing the dataset, implementing the baseline model and the TD-RNN. Performed debugging for the models. Designed the poster.

Ren Hao (not enrolled in the class): Design the TD-RNN with loss function specification. Wrote the report.

VII REFERENCES

E. Hadavandi, H. Shavandi, A. Ghanbari, "Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting", Knowledge-Based System, Vol. 23, No. 8, 2010, pp. 800-808.

A. Fan, and M. Palaniswami, "Stock Selection Using Support Vector Machines", Proceedings of the International Joint Conference on Neural Networks, Vol. 3, 2001, pp. 1793-1798.

K.J. Kim, and W. Lee, "Stock Market Prediction Using Artificial Neural Networks", Neural Computing & Applications, Vol. 13, No. 3, 2004, pp. 255-250.

K.Y., Shen, "Implementing Value Investing Strategy by Artificial Neural Network", International Journal of Business and Information Technology, Vol. 1, No. 1, 2010, pp. 12-22.