PREDICTING STOCK PRICES WITH NEURAL NETWORKS

Hannah Powers
Rensselaer Polytechnic Institute
powerh@rpi.edu

Pierce Phillips
Rensselaer Polytechnic Institute
phillp2@rpi.edu

ABSTRACT

Very general paragraph on what work is done in the paper and some motivation.

1 INTRODUCTION

1.1 BACKGROUND

In many studies and in the world of finance, stock prices have been shown and understood to exhibit the Martingale Property, where their future prices are independent of their past prices. Therefore, if we were to say that stock prices were martingale then historical data and performance would provide no insight or knowledge into the present or future price. But many of these results require an external given rate of return for the market to be martingale, and LeRoy (1973) shows that this property of stock prices is evident when risk-aversion within the market is present, and without this presence trivially predicting stock prices is too difficult and complex. This opens up the question as to whether stock prices truly exhibit being martingale or not.

Another key insight is that stock prices don't follow a fixed path or pattern based on historical data, therefore questioning the reliability of past performance as an indicator of future values. With all the external factors that can influence the price of a stock researchers looked towards the efficient market hypothesis (EMH) as a model to describe the pattern of stock prices. Takeuchi et al. (2011) rejected the hypothesis that stock prices are martingale and can be described by the EMH when tested on some Japanese stocks. Researchers today see a need for a more advanced statistical modeling technique to be able to not only understand the movements of prices but the potential to predict their future performance.

1.2 MOTIVATION AND APPROACH

Time series forecasting has long been a main candidate for statistical modeling within the world of financial returns and prices. Machine Learning has become one of the biggest candidates for modeling these future values, especially within Deep Learning. Even with simpler variables, utilizing Deep Learning models and methods on raw data have shown to be able to construct features that are complex and useful, and shown to be especially good at predicting complex, nonlinear and noisy stock market data (Hoseinzade & Haratizadeh, 2019; LeCun et al., 2015).

In this paper we are going to explore the use of Neural Networks, specifically the Recurrent Neural Network (RNN) architecture, in accurately predicting stock prices based on its historical data. We will develop shallow models to investigate each performance against one another and then will further explore the impact that depth has on these models in individual performance and in comparison with one another. We will then explore a novel approach of using a deep ***insert chosen novel architecture*** for stock price prediction.

2 RELATED WORK

With the ever-increasing use of neural networks in the world, it is no surprise that they are popular for predicting stock prices. Shah et al. (2019) report that the use of machine learning for the field

shows promise and have started to dominate the field over previously used statistical models, such as Auto-Regressive Integrated Moving Average (ARIMA). Siami-Namini et al. (2019) displays the superiority of Recurrent Neural Networks (RNN) over ARIMA, achieving a significantly lower error across all their datasets. Given the time-series nature of stock prices, RNNs are a natural choice of model compared to other methods. Sethia & Raut (2018) notes that both Long Short Term Memory (LSTM) RNNs and Gated Recurrent Unit (GRU) RNNs outperformed both an Artifical Neural Network (ANN) model and a Support Vector Machine (SVM) model. We can also see them outperform linear models in Elliot et al. (2017).

Two other neural architectures that are becoming more popular for sequential data forecasting are Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs). Zhou et al. (2020) show that using a temporal convolutional network not only reduced the time consumption for training and testing but also performed better than RNNs on sequence prediction. One issue that many researchers find is that trying to combine, or stack, multiple different architectures creates challenges to tuning the hyperparameters to achieve reliable results. But Lin et al. (2021) notes that during normal market activity periods a GAN using GRU and CNN layers performed better than LSTM and RNN architectures on stock price prediction.

3 MODELS

Discuss the objective functions and other relevant equations. Include figures of model architectures? Algorithm listings if necessary, performance guarantees, proofs, etc.

All models are trained with an objective function of Mean Squared Error (MSE), that is

$$\min \frac{1}{N} \|\hat{P} - P\|_2^2$$

where $\hat{P}$ are the predicted prices, P are the true prices, and N is the number of instances in the training data. The functions used to calculate $\hat{P}$ for each model are described as follows.

3.1 RECURRENT NEURAL NETWORK (RNN)

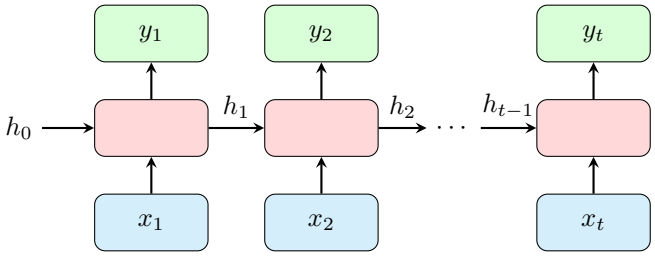

Figure 1: Diagram of unrolled RNN (Gittens, 2023)

An RNN updates its hidden state with the following function

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_t)$$

where σ is an activation function, W_x is the weights applied to the input x, W_h is the weights applied to the hidden state h, and b_t is a constant bias.

3.2 LONG SHORT-TERM MEMORY (LSTM)

LSTM networks were introduced in order to enable an RNN to capture and preserve long-term information and dependencies that is usually lost in RNN's when dealing with sequential data. The design of an LSTM architecture resembles the same structure of the unrolled RNN in Figure 1 above but the difference lies in the structure of the RNN and LSTM cells, as seen in Figure 2 below.

An LSTM cell takes in current input x_t, previous hidden state h_{t-1} and previous cell state c_{t-1}. We can then use the same equation for updating the next hidden state in an RNN to decide on what

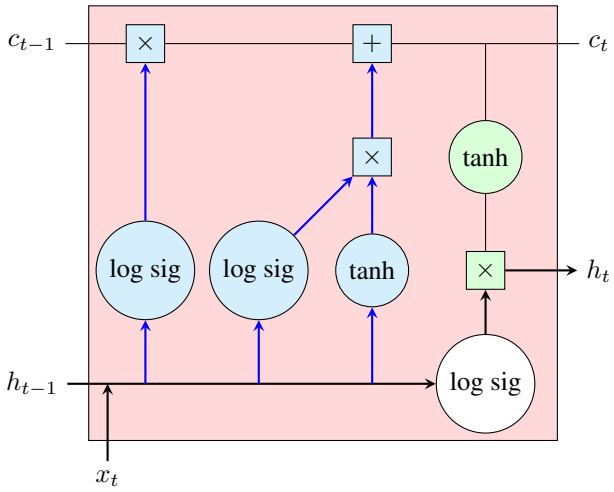

Figure 2: Diagram of LSTM cell (Gittens, 2023)

previous information we should forget and which we should use to update the current cell state through forget and update gates. The outputs of both of these gates are then multiplied element-wise to the the previous cell state respectively and then added together to get the current updated cell state c_t. This cell state is then put through another activation function and the final output gate in order to create the hidden state h_t that is the output, along with c_t, of the current LSTM cell.

3.3 Gated Recurrent Units (GRU)

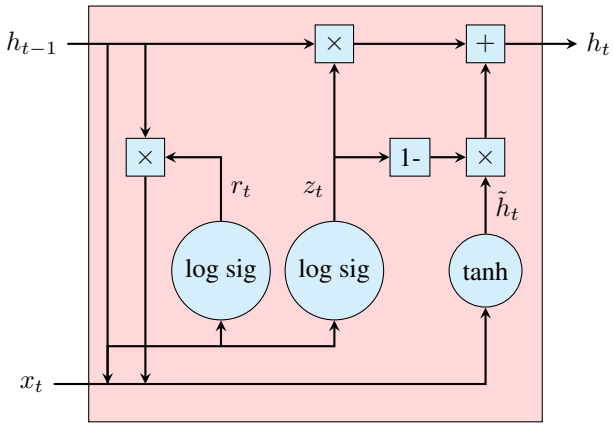

Figure 3: Diagram of GRU cell (Gittens, 2023)

A GRU is also a special kind of RNN whose structure is similar to LSTM with gated mechanisms to control what information is shared between cells. It differs though in having only two gates, an update and reset gate. These gates decide what information should and shouldn't be passed along to the next cell and can be trained to not only remember information from many previous cells but also dispose of information that is not useful for prediction.

Both GRUs and LSTMs solve the vanishing and exploding gradient problems. GRUs though have fewer parameters than LSTMs due to having one less gate and also lack a cell state so they have to store any long and short-term information in its hidden state (Lin et al., 2021).

4 EXPERIMENTS

4.1 DATA

We decided to focus on pulling data on various stock tickers to measure accuracy as well as adaptability of our models in price prediction. We used the closing price only in our analysis as any other variable wouldn't give any additional insight or benefit to our models performance. We are training on the closing price of each ticker from January 1^{st}, 2000 to January 1^{st}, 2020 with a bin size of 14 days where all but the last day is used as the training days and the final day in each bin are the target price we are trying to predict.

4.2 EVALUATION

We use a number of metrics to evaluate the performance of these models. We use both the R-squared and root mean squared error measures from scikit-learn defined as follows:

$$RMSE = \sqrt{\frac{1}{|D|} \sum_{(p,\hat{p}) \in D} (p - \hat{p})^2}$$

and

$$R^2 = 1 - \frac{\sum_{(p,\hat{p}) \in D} (p - \hat{p})^2}{\sum_{(p,\hat{p}) \in D} (p - \overline{p})^2}$$

where p is the true price, $\hat{p}$ is the predicted price, $\overline{p}$ is the mean price of the observed data, D is the dataset, and $|D|$ indicates the size of the dataset. We also use the optimism and pessimism ratios defined in Sethia & Raut (2018).[1] We have

$$OR = \frac{\sum_{(p,\hat{p}) \in D} I(\hat{p} > 1.015p)}{|D|} \quad \text{(Opt Ratio)}$$

$$PR = \frac{\sum_{(p,\hat{p}) \in D} I(\hat{p} < 0.985p)}{|D|} \quad \text{(Pes Ratio)}$$

where $I(\cdot)$ is the indicator function that returns 1 when true and 0 otherwise. Ideally, a good model will have small values for OR and PR.

4.3 RESULTS

	1-Layer			2-Layer			3-Layer		
	RNN	**GRU**	**LSTM**	**RNN**	**GRU**	**LSTM**	**RNN**	**GRU**	**LSTM**
RMSE	0.364	0.072	0.317	0.473	0.038	0.323	0.466	0.058	0.413
R-Squared	-0.304	0.948	0.007	-1.203	0.986	-0.032	-1.140	0.967	-0.680
OR	0.110	0.116	0.005	0.001	0.677	0.060	0.045	0.780	0.0280
PR	0.969	0.955	0.998	0.999	0.398	0.963	0.985	0.263	0.993

Figure 4: Model results on the test data for the metrics defined in Section 4.2

Once all models are trained on the data, we will test them and tabulate their values for the metrics defined in Section 4.2. In addition, we hope to eventually show the returns of the model via one or more trading strategies so we can examine practical performance for the short- and long-term.

5 CONCLUSION

After the results have been gathered, we will compare the models based on their accuracy shown with the evaluation metrics. We would also like to draw conclusions on which model would be most effective for trading.

[1] Code for our research project can be found at https://github.com/hpowers57/MLOpt_ResearchProject.

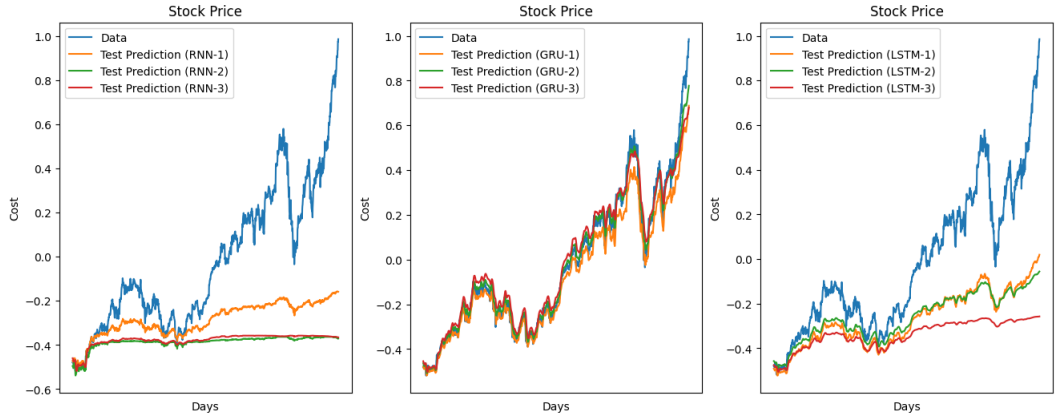

Figure 5: The actual vs. predicted stock price for all nine model types with varied layers

6 Future Work

Our future work will depend on the work we can accomplish for this project. If we are unable to get to it, then it will at least include an examination of trading strategies and the best model for each.

References

Aaron Elliot, Cheng Hsu, and Jennifer Slodoba. Time series prediction: Predicting stock price. 2017.

Prof. Alex Gittens. Lecture notes for machine learning and optimization, April 2023.

Ehsan Hoseinzade and Saman Haratizadeh. Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285, 2019.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

Stephen F. LeRoy. Risk aversion and the martingale property of stock prices. *International Economic Review*, 14:436–446, 1973.

HungChun Lin, Chen Chen, GaoFeng Huang, and Amir Jafari. Stock price prediction using generative adversarial networks. *Journal of Computer Science*, 17:188–196, 2021.

Akhil Sethia and Purva Raut. Application of lstm, gru and ica for stock price prediction. *International Conference on ICT for Intelligent Systems*, 2:479–487, 2018.

Dev Shah, Haruna Isah, and Farhana Zulkernine. Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies*, 7:26, 05 2019.

Sima Siami-Namini, Neda Tavakoli, and Akbar S. Namin. The performance of lstm and bilstm in forecasting time series. *IEEE International Conference on Big Data*, pp. 3285–3292, 2019.

Kei Takeuchi, Akimichi Takemura, and Masayuki Kumon. New procedures for testing whether stock price processes are martingales. *Computational Economics*, 37:67–88, 2011.

Kun Zhou, Wenyoung Wang, Teng Hu, and Kai Deng. Time series forecasting and classification models based on recurrent with attention mechanism and generative adversarial networks. *Sensors*, 20:7211, 2020.