Predicting Stock Prices with Neural Networks

Pierce Phillips & Hannah Powers

CSCI 6968 2023

Introduction

In the world of finance, many theories have come about on the trends of stock prices. One theory considers the Martingale Property of stocks: that for a sequence of random variables, the expected value of the next value in the sequence given the previous values is the current value. For stock prices, this means that historical data will provide little insight in the future price. Another theory used by researchers, the efficient market hypothesis (EMH), proposed that all new information was immediately reflected in the stock price. But this approach for forecasting only works under the assumption of a risk-neutral market which doesn't readily exist in the real-world markets.

Motivation and Related Work

Machine learning has proven it is capable of learning exceedingly complex trends and data and it's no surprise it has it's applications in the world of finance. Given the sequential nature of stock price data, the use of recurrent neural networks in the field is only natural, taking place over previously preferred statistical models such as Auto-Regressive Integrated Moving Average (ARIMA) [1]. We see RNNs beat out ARIMA in [2] and even beat out other ML and linear models in [3] and [4]. Our goal is to continue this work and examine the effect of model depth on performance as well as develop a robust method that can outperform existing models.

Models

We examine five types of models for our research:

- Martingale model
- Recurrent Neural Network (RNN)
- Long-Short Term Memory network (LSTM)
- Gated Recurrent Unit network (GRU)
- And our novel model: long-short Gated Recurrent Unit (ls-GRU)

All models have an objective function of Mean Squared Error (MSE):

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

with true prices P, predicted prices $\hat{P}$, and N as the number of datapoints

Martingale

The Martingale model is derived from the concept of martingales. We can let x_t be a random variable at time t that follows from a sequence $S = \{x_0, x_1, x_2, ...\}$. The property of martingales states that

$$\mathbb{E}[x_t | x_0, x_1, ..., x_{t-1}] = x_{t-1}$$

Thus, the Martingale model, when provided a sequence of input, predicts the last value of the sequence to be the next value [4]

Recurrent Neural Network (RNN)

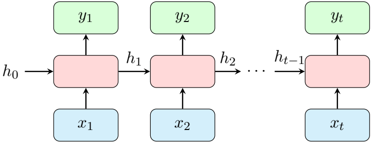

Figure: Diagram of unrolled RNN [5]

RNNs are made to take advantage of the sequential nature of certain data. The hidden weights of the model are updated with each input and fed back into the cell for the next input in the sequence. We update with

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

where σ is either the logistic sigmoid or tanh activation functions, W_x is the weights on input x_t, W_h is the weights on hidden layer h_{t-1}, and b_t is the bias term

Long-Short Term Memory (LSTM)

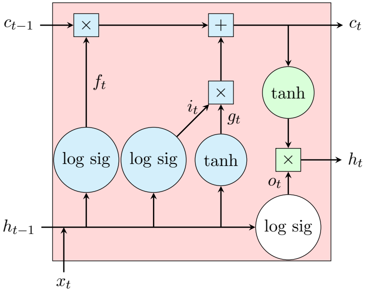

Figure: Diagram of LSTM cell [5]

LSTMs have a similar recurrent nature to RNNs but a more complex cell structure with four gates: input gate i_t, forget gate f_t, cell gate g_t, and output gate o_t. It also has an additional cell state variable c_t which is passed back into the cell with h_t.

Gated Recurrent Unit (GRU)

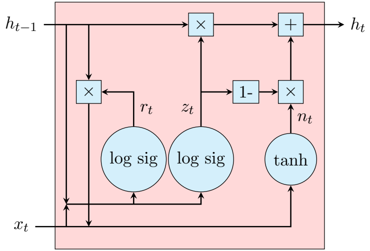

Figure: Diagram of GRU cell [6]

Like LSTMs and RNNs, GRUs feed their hidden state back into the cell for the next input. A GRU cell only has three gates as compared to an LSTM: reset gate r_t, update gate z_t, and new gate n_t, and it doesn't have a cell state.

Long-Short Gated Recurrent Unit (LS-GRU)

Figure: Diagram of LS-GRU

Our novel model combines several GRU and LSTM models: we stack 3 GRUs onto 2 LSTMS and 3 more GRUs. The output of each model is fed in as the input for the next model. The goal is combine the GRU's capabilities in closely following trends in the data with an LSTM's tendency to overpredict to create a robust model.

Data

Our data is downloaded from Yahoo Finance. We use closing price only as an examination of the EMH.

- Training data: Apple Inc (AAPL) from 01/01/2000 to 01/01/2017
- Validation data: Apple Inc (AAPL) from 01/01/2017 to 01/01/2020
- Testing data: Wells Fargo (WFC), Tesla (TSLA), General Mills (GIS), and American Tower Corp (AMT) from 2000 to 2020

Data is pre-processed as follows:

1. Min-max scaling with scikit-learn so all values lie in [-1,1]
2. Binned on every two weeks
3. Last day in the two week period represents the outcome, feature space $x \in \mathbb{R}^{13}$ and outcome $y \in \mathbb{R}$

Evaluation

All models are evaluated on the following four measures:

- Root Mean Squared Error (RMSE) = $\sqrt{\frac{1}{|D|}\sum_{(p,\hat{p})\in D}(p-\hat{p})^2}$
  - An accurate model will have values near 0
- $R^2 = 1 - \frac{\sum_{(p,\hat{p})\in D}(p-\hat{p})^2}{\sum_{(p,\hat{p})\in D}(p-\overline{p})^2}$
  - Values typically range from 0 to 1 with 1 being most accurate and values may go to the negatives for very poor models
- Optimism Ratio = $\frac{\sum_{(p,\hat{p})\in D} I(\hat{p}>1.015p)}{|D|}$ [3]
  - Ideal models will have OR near 0, high values means the model overpredicts
- Pessimism Ratio = $\frac{\sum_{(p,\hat{p})\in D} I(\hat{p}<0.985p)}{|D|}$ [3]
  - Models should have PR near 0, high values means model underpredicts
  - An accurate model will have PR and OR near 0

Results

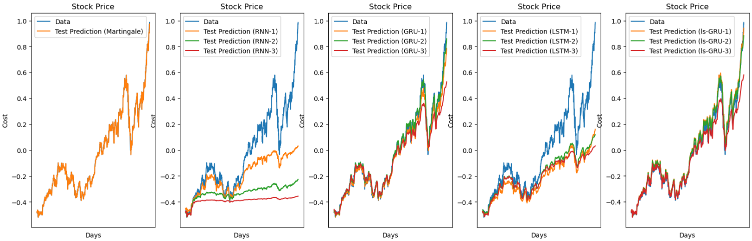

Figure: Actual v. Predicted stock price for all models with varied layer counts AAPL stocks

Visually, we see that no model seems to outperform the Martingale model however the 2-layer GRU and 2- and 3-layer ls-GRU come very close. These models are also much better at following the trends of the data than RNN and LSTM.

Results

		RMSE	R-Squared	OR	PR
	Martingale	**0.015912**	**0.997501**	0.525896	0.600266
1-Layer	**RNN**	0.255759	0.35445	0.102922	0.934263
	GRU	0.044402	0.980543	0.233732	0.87251
	LSTM	0.255294	0.356797	**0.035193**	0.995352
	ls-GRU	**0.034726**	**0.988099**	0.923639	**0.126826**
2-Layer	**RNN**	0.410256	-0.661027	**0.051129**	0.995352
	GRU	**0.030074**	**0.991074**	0.710491	**0.363214**
	LSTM	0.235926	0.450689	0.094954	0.962151
	ls-GRU	**0.032112**	**0.989823**	0.870518	**0.204515**
3-Layer	**RNN**	0.47496	-1.226288	**0.02988**	0.996016
	GRU	0.088512	0.922684	0.425631	0.661355
	LSTM	0.26359	0.314317	**0.066401**	0.971448
	ls-GRU	0.073291	0.946989	0.438247	0.726428

Table: Model results on the validation dataset, top 4 models are bolded for each measure

Results

- Martingale model has the best accuracy of all, followed by the 2-layer GRU and then the 3- and 2-layer ls-GRUs which supports the graphs earlier
- The ls-GRU models perform better in general than the other architecture types
- GRUs are a close second in accurate architectures to ls-GRUs
- No model is able to achieve both a low OR and PR but the Martingale, 3-layer GRU, and 3-layer ls-GRU almost balance the ratios, not skewed toward over- or under-predicting prices
- GRUs were best at balancing their OR and PR values

Conclusion

As we had hoped, the ls-GRU was a more robust model type than GRUs and LSTMs alone. It had notably better performance in accuracy. That the Martingale model beat out all other models is a mark in favor of the Martingale Property of stocks in the short-term. In the long-term with more shocks to the stock price and the market as a whole, the robustness of the neural networks outperformed the more trivial martingale process.

References I

- D. Shah, H. Isah, and F. Zulkernine, "Stock market analysis: A review and taxonomy of prediction techniques," *International Journal of Financial Studies*, vol. 7, p. 26, 05 2019.
- S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of lstm and bilstm in forecasting time series," *IEEE International Conference on Big Data*, pp. 3285–3292, 2019.
- A. Sethia and P. Raut, "Application of lstm, gru and ica for stock price prediction," *International Conference on ICT for Intelligent Systems*, vol. 2, pp. 479–487, 2018.
- A. Elliot, C. Hsu, and J. Slodoba, "Time series prediction: Predicting stock price," *arXiv*, 2017.

References II

P. A. Gittens, "Lecture notes for machine learning and optimization," April 2023.

A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*, ch. 10.
arXiv, 2021.