CS230

Generative Adversarial Network for Stock Market price Prediction

Ricardo Alberto Carrillo Romero

Stanford University
racr@stanford.edu
SUNet ID: 06409645

Abstract

This project addresses the problem of predicting stock price movement using financial data. Although the extensive exploration with GAN, we found that the relative performance of GAN model with respect traditional deep learning models such as LSTM has not been assessed. Our specific goal was to predict whether the price would increase one day after our sample period. We trained a baseline ARIMA model, a long short-term memory (LSTM) model, a deep LSTM model, and a generative adversarial network (GAN) model to develop this task. All our models predicted near or above 60% accuracy. The best performer were the Shallow LSTM 74.16% and the GAN 72.68 % and the Deep LSTM 62.85% followed by ARIMA 59.57%.

1 Introduction

Stock price prediction in capital markets has been consistently researched using deep learning, just last year, there were at least 9700 papers written on the subject according Google Scholar. Related to Time Series, recurring neural networks such as long short-term memory (LSTM) had been successfully tested to replicate stock price distributions. Similarly, since 2014, generative adversarial networks (GAN) have also been the subject of intense experimentation, especially in the field of computer vision. We explored using a GAN with a convolutional neural network (CNN) as a discriminator and multi-layer perceptron as a generator to forecast the closing price of stocks. We took as an input 20 days of price data and financial indicators. In addition, we experimented with the fundamental data of the companies. More formally, given a series of financial data for X1, X2, X3...X20 for a stock, we attempted to predict whether X21>X20 was true.

2 Related work

Stock market prediction is widespread via time series models (e.g., ARIMA, ARIMA with SVM, CNN , LSTM (1), attentive neural models (2)). For example, in 2018, Sima Siami and Akbar Siami (3) compared price forecasting using ARIMA and LSTM and had promising results with LSTM, more specifically, the average reduction in error rates obtained by LSTM was between 84 - 87 percent when compared to ARIMA indicating the superiority of LSTM to ARIMA. Meanwhile, Kang Zhang et al.(4) proposed a GAN architecture with an LSTM generator and multi-layer perceptron as a discriminator to predict the closing price of the Standard and Poor Index, Shangai Composite Index, International Business Machines, and Microsoft stocks. The datasets used involved seven financial factors and the results of showed a GAN MAE of 3.04 vs LSTM MAE of 4.12.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

We found inspiration from those studies to explore the use of a GAN model to represent the data distribution of a stock price and then predict the movement of the stock one day in the future.

3 Dataset and Features

As previously stated, the input of the models in this project are price data and financial indicators.

Source. We used Alpha Vantage (5) for our GAN model. For ARIMA and for our LSTM models, we obtain the data through the Sharadar Core US bundle provided by Quandl (6) and specifically implementing the following tables: Sharadar's Equity Prices, Daily Metrics, and Core US Fundamentals. These provided daily open/close prices, enterprise value over earnings, price-to-book value (pb), price-to-earnings value (pe) and volume.

Structure. Each example $x^{(i)}$ consist a input $x^{(t)} \in \mathbb{R}^{(20x5)}$ which represents 20 daily values of the following daily price data: open price, high price, low price, close price, and volume. In our LSTM, we experimented with two additional datasets with more features: one with 13 features including price data and financial indicators such as enterprise value, enterprise value over EBIT, enterprise value over EBITDA, market capitalization, pb, pe, and price sales (seven values from Sharadar Equity prices). In our GAN, we tested the data over 500 Standard and Poor's companies.

Preprocessing. We normalized all the data for the final file of a stock and then merged, dropped columns, and completed the data of the three Sharadar tables to extract and compile the data that fed the model into one file.

Training/validation/test split. Our data came from 2014 and included 1,400+ daily samples of each stock. We split the training/validation/test sets into 90/5/5. Our dataset for training the GAN model involves more than 370.000 samples.

Labeling. We labeled y(i) d, where i represents the price of the stock in d future days. Our original approach was d = 5, but after several experiments, we decided to set the future predicted value at d = 1.

Normalization We normalized our data set before training so that our whole data features has zero mean and unit variance. Specifically, for a feature in our entire dataset X1, X2. . . Xn, we computed the normalized prices, Xi=Xi-X/Sx, where X and S are the mean and standard deviation respectively,we used Xi as the input to the model instead.

4 Methods

This section discusses the architecture and parameters of the methods used. For comparison purposes, the methods include the classic ARIMA and those based in deep learning architectures.

4.1 Baseline. We experimented with the statistical model ARIMA for forecasting the stock price time series. We used the model with the following characteristics: five lag value and stationary and moving average of zero. We implemented the ARIMA model in Python with the StatsModels package.

4.2 Shallow Long Short term Memory. We decided to use the architecture shown in Figure 1, using one LSTM layer with 32 hidden units. The final activation was a one-unit dense layer with a linear activation. We also experimented with several hidden units with the LSTM and obtained the best results with the 32 hidden units. Our LSTM models were implemented in TensorFlow(7) and KERAS(8).

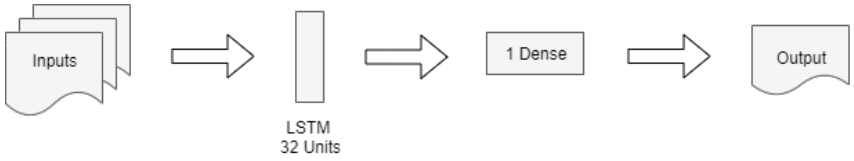

Figure 1

4.3 Deep Long Short-Term. The architecture for the deep LSTM consisted of two LSTM layers with 32 and 16 hidden units concatenated with three fully connected layers. The first three dense layers used tanh activation and the final dense unit a linear activation. We experimented with several values for the hidden layers in the LSTM. MAE was used as loss function, after the training are able to predict if the stock will go up in the 21 day a the sequence.

Figure 2

4.4 Generative Adversarial Network Model. The GAN architecture used has a three-layer dense network as a generator and a three-layer CNN as the discriminator. The discriminator was implemented with three convolutional layers, followed by a flattened layer and one dense layer with sigmoid activation.

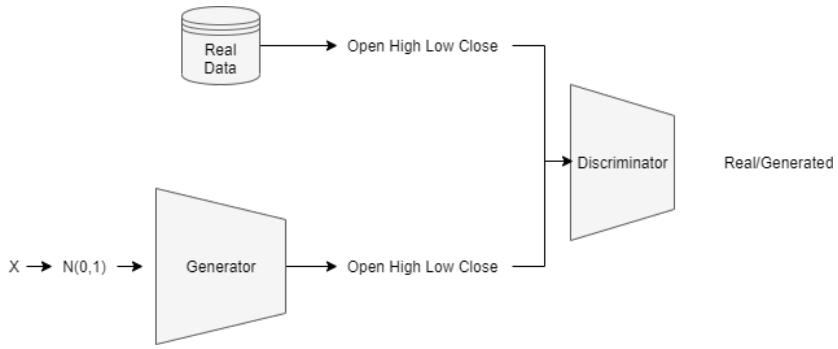

Figure 3

5 Experiment/Results/Discussion

5.1 Model Results

Table 1: Summary of model accuracy for Up Prediction

Model	Features	Train Acc.	Validation Acc.	Test Acc.
ARIMA	N/A	N/A	59.16%	N/A
Shallow LSTM	4	59.95%	62.02%	74.16%
Deep LSTM	4	79.26%	88.35%	62.85%
GAN	5	73.04%	72.16%	72.68%

5.2 Baseline Analysis. The baseline model performed comparably to the deep learning models. The ARIMA model reached about 59.16% accuracy in the validation set.

5.3 Long Short-term Memory Analysis.

5.3.1 Evaluation and Hyperparameters. We conducted hand tuning of multiple parameters but in a random manner [6]. Our LSTM architectures used a regression loss function, specifically, mean absolute error (MAE), with the purpose of making the models more robust against outliers. After we trained our Shallow LSTM model during 50 epochs, we classified the forecast to obtain our acurracy metrics.

We noted that smaller batches improved our results, so we select a batch size of 128. We selected the Adam method as an optimizer after testing the RMSProp and stochastic gradient descent with learning rate of 0.01 and rate decay of 1e-6 and 1e-5, the best learning rate for our Adam optimizer was 1e-3. In the deep LSTM, we L1 regularization in the two LSTM layers, because we were working with the MAE error.

The Deep LSTM model had a particularly unfavorable behavior whit increments in the hidden units beyond 64 or 128, we believe that we need a bigger set of features to handle more hidden units in LSTM. For experimental purposes we built 4 different datasets with 4,13, 54 and 109 features to explore performance of Shallow and Deep LSTM with individual stocks data.

Figure 4.Shallow LSTM Training and Validation Loss

5.3.2 Long Short-term Memory Performance. Shallow LSTM shows a better accuracy than Deep LSTM, additionally we found that the Deep LSTM tend toward to predict a increase in price, for which we hypothesize the follow: Historically stocks prices tend to increase, specially in the latest years.In consequence, the models may actually be biased by the unbalanced label distribution (55-45) in favor of an increase.

5.4 Generative Adversarial Network Analysis

5.4.1 Generative Adversarial Network Evaluation and Hyperparameters We experimented using the GAN model with 20K, 30K, and 50K epochs, obtaining our best results in the 50K epoch value. We also experimented with forecasting the future in one, two, and five days. In terms of the optimization algorithm, we used the Adam optimizer with the best value in 1e-4. Ultimately, we achieved our best results with 50k epochs and prediction one day ahead.

5.4.2 Generative Adversarial Network Performance As can be seen in the confusion matrix, the predicted up price was `72.68%`. This was slightly better than the prediction of the decrease of a price.

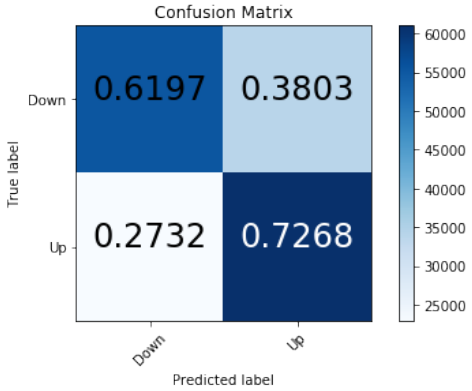

Figure 5

6 Conclusion/Future Work

We can make at least two relevant conclusions: The first is GAN network architecture can be used to make representations of time series and specifically representations of the distribution of the stock price of capital markets. Our second conclusion is at least in the deep learning models we used, there are no significant differences with GAN and models traditionally used as LSTM. Another finding is the fact that more effort was required for the tuning of models with many features.

The following experiments could be carried out in the future to further develop the project:

- Use of different architectures for the GAN. Explore different GAN architectures to simulate time series, especially those that involve structures traditionally used for time series in deep learning such as LSTM.
- GAN loss and tuning mechanisms. Explore loss functions different from traditional ones with GANs, such as WGAN, which uses Wasserstein distance(9), and explore whether the tuning of these networks can be improved via reinforcement learning.
- Extend the use of GAN for better distribution selection. Experiment with Metropolis-Hastings GAN (MHGAN)(10), where after training the generator, instead of discarding the discriminator use it to select the distribution closest to the actual distribution created by the generator.

Github Repository Link

All code written for this report can be found in the following Github repository: `https://github.com/carrilloric/CS230`

References

[1] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10614 LNCS, pp. 729–730, 2017.

[2] "ATTENTATIVE NEURAL MODELS FOR ALGORITHMIC TRADING REPORT.pdf."

[3] S. Siami-Namini and A. S. Namin, "Forecasting Economics and Financial Time Series: ARIMA vs. LSTM," pp. 1–19, 2018.

[4] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock Market Prediction Based on Generative Adversarial Network," *Procedia Computer Science*, vol. 147, pp. 400–406, 2019.

[5] AlphaVantage, "Stock Time Series time series intraday," 2016.

[6] Quandl, "WIKI various end-of-day data," 2016.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[8] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.

[9] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *34th International Conference on Machine Learning, ICML 2017*, 2017.

[10] R. Turner, J. Hung, E. Frank, Y. Saatci, and J. Yosinski, "Metropolis-Hastings Generative Adversarial Networks," 2018.