GAN base 7-minute prediction in stock market

Daehan Han
Department of Computer Science
Stanford University
CS230
daehan.han78@gamil.com
deahanh@stanford.edu

Abstract

The prediction of stock prices has been attempted many times without much success. However, the prediction of long term stock prices is difficult to access because it requires a lot of information for analysis, and the surrounding environment dramatically influences it. Therefore, short term stock prediction, which is less susceptible to the surrounding environment, is the subject of research. The PJT challenged the stock price forecast through the Generative Adversarial Network (GAN) model. Stock price prediction is approached by people who have learned financial engineering based on various methods. It is difficult for the general public to analyze and predict the stock trend. So I tried to apply the trader's propensity through GAN and apply it to short term stock predictions.

1 Introduction

In the case of Long Short-Term Memory(LSTM), this model generally used for time series prediction. It is possible to predict when there is a given condition, but it may be difficult to predict when a new input provided. To overcome these problems, I use the Generative Adversarial Network (GAN) based on the prediction model. In this research, I apply for the LSTM as a generator and a Convolutional neural network (CNN) as a discriminator [3]. The input data is the price and trading volume, plus the data based calculated technical indicators and ignores the surrounding environment for the stock market. Various types of hyper parameter splits draw optimal conclusions.

2 Related work

The previous research direction on stock price prediction was 'Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets', which is proposed by ‘Xingyu Zhou'. [3] This approach was to predict stock price after 1 minute by implementing Generative Adversarial Network. That was introduced by ‘Goodfellow et al.'[2]. However, the purpose of this research is to predict the result of stock market prices 7 minutes later, rather than 1 minute. The evaluation metric compared to apply the weight of each time step rather than Root Mean Squared Relative Error (RMSRE) [3].

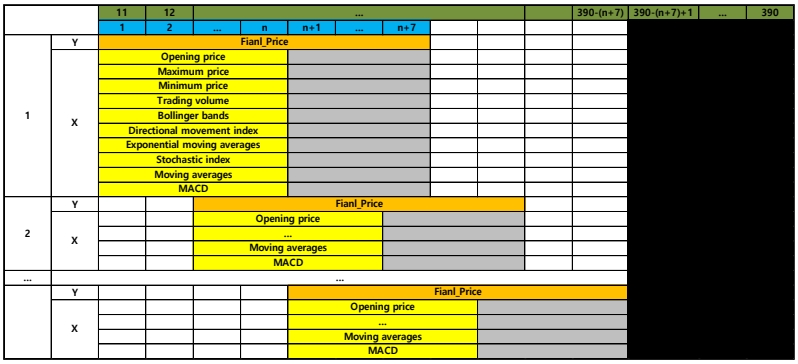
3 Dataset

Data were analyzed based on the same technical indicators of stocks as in the previous study.[3] I won't go into specific mention of that metric, but when you make indicators, zero denominators can happen, so you must take care of these problems. Total count for data is 16560, and randomly shuffled and divided train, validation, and test set as 14500,1035, 1025.

The technical indicators used in the data set are as follows.

Indicators
Opening price
Maximum price
Minimum price
Trading volume
Bollinger bands
Directional movement index
Exponential moving averages
Stochastic index
Moving averages
MACD

The Korean stock market starts at 9:00 and closes at 15:30, so there are 390-time steps for 1-minute intervals. In the first 10 minutes, data cannot use due to calculating the indicators. Besides, I only collect the daily data, which is only a variation of day more than 20%.



4 Method

The prediction Model applied GAN-FD model, and the detailed shape explained in the following figure. The LSTM model is used to generate the fake result $\hat{Y}$ (generated result is final price), and the CNN model determines whether the generated fake stock price is real or fake.

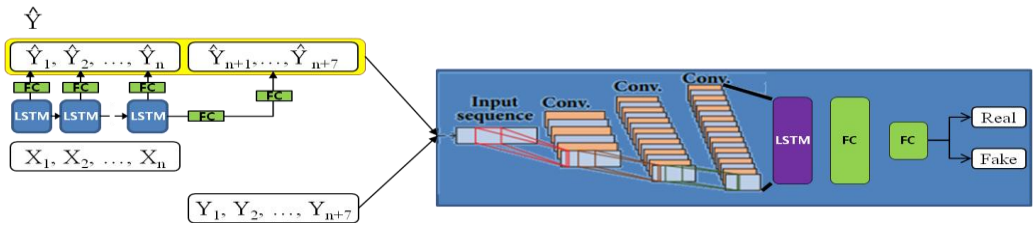


Figure 1: GAN-FD architecture. Te generator (G) is founded on LSTM, which applies to predicting $\hat{Y}$. Te discriminator (D) is based on CNN for the purpose of estimating the probability whether a sequence is real (Y) or being predicted ($\hat{Y}$). Conv. means the convolutional layer, FC is an abbreviation for fully connected layer. Te structure of G and D can be adjusted according to the specific application.[1]

4.1 Loss Function

The test conducted to apply the unconditional GAN loss function. When the train performed with the basic GAN loss function, the sensitivity of the hyper parameter was too big. Moreover, the train uniformity of the generator was not reasonable. Additionally, in this result, G could generate samples to "confuse" D, without being close to the ground truth.

Normal GAN Loss function

$$DiscriminatorLoss = \min_{G}\max_{D} V(D,G) = \mathbb{E}_{y\sim p(y)}[logD(y)] + \mathbb{E}_{y_{fake}\tilde{} p(y_{fake})}[log(1 - D(G(y_{fake})))]$$

$$GeneratorLoss = \mathbb{E}_{y_{fake}\tilde{} p(y_{fake})}[log(1 - D(G(y_{fake})))]$$

Therefore, the mean absolute error added to the loss function, which was introduced by previous research [3]. The result was improved, but the accuracy of the result far away forms the ground truth. So I added one more thing to improve the prediction accuracy, which is absolute value for the last value and ground truth.

Final Loss

$$DiscriminatorLoss = \nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[logD(y^{(i)}) + log(1 - D(G(x^{(i)}) = \hat{y}^{(i)})\right]$$

$$GeneratorLoss = \nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[log(1 - D(G(x^{(i)}) = \hat{y}^{(i)}) + W_1 * |y^{(i)} - \hat{y}^{(i)}| + W_2 * |{y_T}^{(i)} - {\hat{y}_T}^{(i)}|\right]$$

4.2 Evaluation metric and Network Architecture

It is essential to predict all the time step of stock price predictions. Nevertheless, the most critical point is to predict a specific future stock price. Therefore, the accuracy comparison of the expected value evaluated the test consistency by giving different weights at each time point. Thus, its method applies to more weight in future forecasts rather than past values.

Evaluation Metrics :

$$TimeWeightedMeanAbsoluteError = \sum_{i=1}^{T}\left[\frac{i}{T}|y^{(i)} - \hat{y}^{(i)}|\right]$$

$$Lastpredictionvaluecompare = |y^{(T+7)} - \hat{y}^{(T+7)}|$$

Network Architecture

[Generator]

```
Gen_LSTM(
  (lstm): LSTM(18, 100, num_layers=2)
  (regressor1): Sequential(
    (0): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (1): Dropout(p=0.0, inplace=False)
    (2): Linear(in_features=100, out_features=50, bias=True)
    (3): ReLU()
    (4): Linear(in_features=50, out_features=1, bias=True)
  )
  (regressor): Sequential(
    (0): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (1): Dropout(p=0.0, inplace=False)
    (2): Linear(in_features=100, out_features=50, bias=True)
    (3): ReLU()
    (4): Linear(in_features=50, out_features=6, bias=True)
  )
)
```

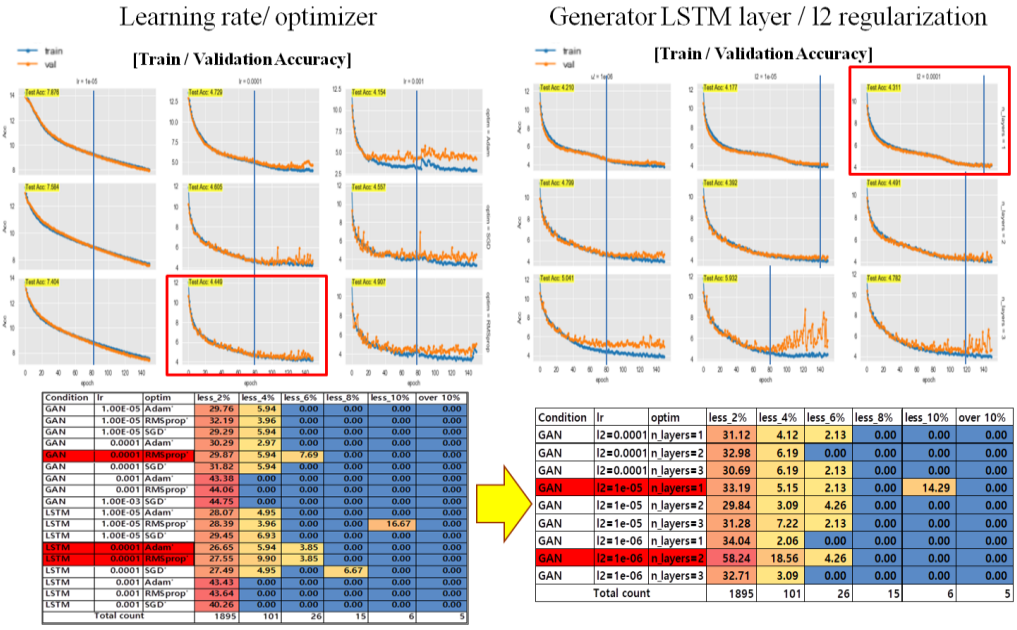
[Discriminator]

```
Dis_CNN(
  (conv1): Sequential(
    (0): Conv1d(1, 32, kernel_size=(3,), stride=(1,))
    (1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (conv2): Sequential(
    (0): Conv1d(32, 64, kernel_size=(3,), stride=(1,))
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (conv3): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,))
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (lstm): LSTM(128, 128, num_layers=2)
  (fc1): Sequential(
    (0): Linear(in_features=128, out_features=64, bias=True)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (fc2): Sequential(
    (0): Linear(in_features=64, out_features=32, bias=True)
    (1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (fc3): Sequential(
    (0): Linear(in_features=32, out_features=1, bias=True)
    (1): Sigmoid()
  )
)
```

5 Experimental/Result/Discussion

The first test shows that the LSTM model is more accurate than the GAN models. After the hyper parameter optimization, the GAN model was enhanced and got a similar accuracy form the LSTM model. The hypothesis in this experiment was that the GAN model would improve over the LSTM, but the lack of training set, the discriminator could not learn enough for the decision. As shown in the figure below, the optimization of each hyper parameter is improved the prediction result. Therefore, we will try to train by adding more data to overcome the above problems in the future research. Moreover, time interval base split will test.

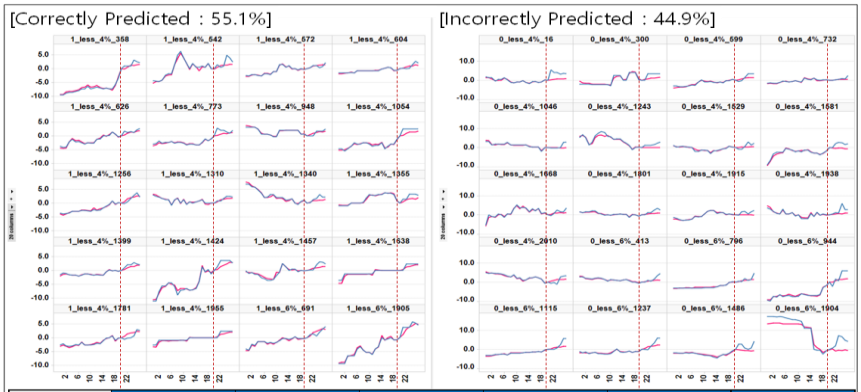
➔ Compare the price variation for the last prediction accuracy [%]



Condition	lr	optim	less_2%	less_4%	less_6%	less_8%	less_10%	over 10%
GAN	1.00E-05	Adam	29.76	5.94	0.00	0.00	0.00	0.00
GAN	1.00E-05	RMSprop	32.19	3.96	0.00	0.00	0.00	0.00
GAN	1.00E-05	SGD	29.29	5.94	0.00	0.00	0.00	0.00
GAN	0.0001	Adam	30.29	2.97	0.00	0.00	0.00	0.00
GAN	0.0001	RMSprop	29.87	5.94	7.69	0.00	0.00	0.00
GAN	0.0001	SGD	31.82	5.94	0.00	0.00	0.00	0.00
GAN	0.001	Adam	43.38	0.00	0.00	0.00	0.00	0.00
GAN	0.001	RMSprop	44.06	0.00	0.00	0.00	0.00	0.00
GAN	1.00E-03	SGD	44.75	0.00	0.00	0.00	0.00	0.00
LSTM	1.00E-05	Adam	28.07	4.95	0.00	0.00	0.00	0.00
LSTM	1.00E-05	RMSprop	28.39	3.96	0.00	0.00	16.67	0.00
LSTM	1.00E-05	SGD	29.45	6.93	0.00	0.00	0.00	0.00
LSTM	0.0001	Adam	26.65	5.94	3.85	0.00	0.00	0.00
LSTM	0.0001	RMSprop	27.55	9.90	3.85	0.00	0.00	0.00
LSTM	0.0001	SGD	27.49	4.95	0.00	6.67	0.00	0.00
LSTM	0.001	Adam	43.43	0.00	0.00	0.00	0.00	0.00
LSTM	0.001	RMSprop	43.64	0.00	0.00	0.00	0.00	0.00
LSTM	0.001	SGD	40.26	0.00	0.00	0.00	0.00	0.00
Total count			1895	101	26	15	6	5

Condition	lr	optim	less_2%	less_4%	less_6%	less_8%	less_10%	over 10%
GAN	l2=0.0001	n_layers=1	31.12	4.12	2.13	0.00	0.00	0.00
GAN	l2=0.0001	n_layers=2	32.98	6.19	0.00	0.00	0.00	0.00
GAN	l2=0.0001	n_layers=3	30.69	6.19	2.13	0.00	0.00	0.00
GAN	l2=1e-05	n_layers=1	33.19	5.15	2.13	0.00	14.29	0.00
GAN	l2=1e-05	n_layers=2	29.84	3.09	4.26	0.00	0.00	0.00
GAN	l2=1e-05	n_layers=3	31.28	7.22	2.13	0.00	0.00	0.00
GAN	l2=1e-06	n_layers=1	34.04	2.06	0.00	0.00	0.00	0.00
GAN	l2=1e-06	n_layers=2	58.24	18.56	4.26	0.00	0.00	0.00
GAN	l2=1e-06	n_layers=3	32.71	3.09	0.00	0.00	0.00	0.00
Total count			1895	101	26	15	6	5

Figure 2: GAN learning rate & optimizer split test result and LSTM layer count & L2 regularization split result. Less 2% mean last ground truth result is under 2% and acc table mean 'compare the last ground truth vs. prediction result.



	less_2%	less_4%	less_6%	less_8%	less_10%	over 10%
Real Count	1880	97	47	9	7	8
Prediction	1095	18	2	0	0	0
	58.2	18.6	4.3	0.0	0.0	0.0

Figure 3: Correctness of the prediction result. Current result shows that 96% of incorrectness. However, if the training size is big enough it will be overcome the correctness.

Acknowledgments

I would like to thank the CS230 teaching team. Sepcial thanks to my advisor Sarah Ciresi. Her support gives me the correct direction for my project and I understand the Generative Adversarial Network more.

References

[1] Code reference https://github.com/borisbanushev/stockpredictionai#thediscriminator

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in Proceedings of the 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014, pp. 2672–2680, can, December 2014.

[3] Xingyu Zhou, Zhisong Pan, Guyu Hu et al., (2018) Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets., Mathematical Problems in Engineering Volume 2018, Article ID 4907423, 11 pages.