Financial Big Data Project

Lin Gao, Shengzhao Lei

January 16, 2019

Abstract

Our project analyzed data of stock market. After data mining we used mean-variance and minimum variance analysis on log return data with different covariance matrix, using both in-sample and out-of-sample methods. In addition, we applied LSTM model on Credit Suisse and Novartis to predict their closing price.

1 Introduction

Stock is a tradition financial instrument appearing hundreds years ago. Since the very beginning, people were exploring various methods to predict the future stock values. As the development of quantitative methods and computing power of modern computers, big data plays a increasingly important role on stock.

By studying and evaluating past and current data, we can make more informed decisions with the help of different models. In this report, we studied both traditional strategy like mean-vaniance portfolio and novel method like LSTM from deep learning.

2 Data Preprocessing

2.1 Overview of Dataset

We collected our data from Yahoo finance (`https://finance.yahoo.com`). More specifically, We firstly found all stock tickers names and then used `fix_yahoo_finance` API to download history data from `2000-01-01` to `2019-01-10` of all stocks. So we had in total 106328 tickers finally. As the amount of data is huge, we downloaded the data 5000 days by 5000 days and saved each chunk as `.csv` file. We then performed concatenation and got the *closing price* series of each stock. In order to import and store data easier, we saved our data as pickle file.

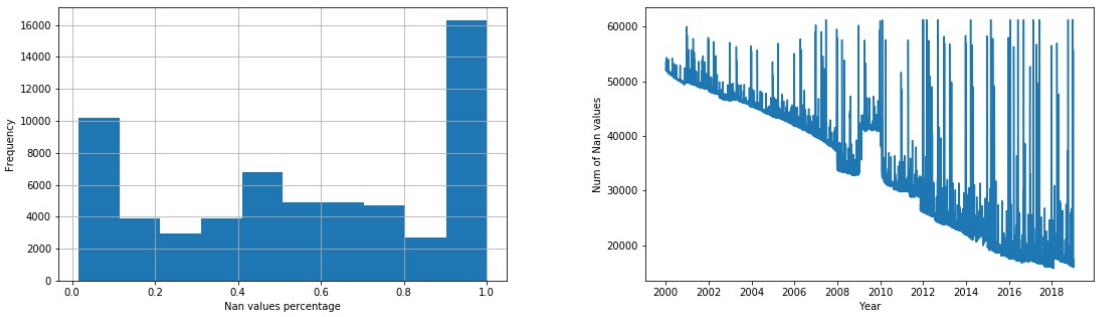

Figure 1: Distribution of missing values

We found that our data contains the price of `Sundays`(which actually are NaN values). Normally there won't be any transaction on weekends, so we chose to remove it and only kept data of weekdays. In addition, we transferred the type of date in our data from `string` to `datetime` which brought lots of convenience in our future analysis. After those processing, we displayed the distribution of missing values by tickers and by years respectively as Figure 1.

2.2 Data Selection and Fill NaN Values

Considering that there are 4,980 samples (days) per stock, we screened out stocks with a missing percentage below 0.025. So, we finally got 454 stocks left. With meta information, we found they belong to 10 exchanges, of which distribution is shown as Figure 2. Most stocks are from LSE and PAR.

We used `ffill` method which fills missing values with forward values. Then, we dropped the stocks which have NaN values in the first several days since there are no previous values of them. After this, we got our final *closing price* data, which has 4980 days and 412 stocks.

Finally, we transformed *closing price* to *log-return*, which is the data we will use for future analysis. We report the log-return of Credit Suisse Group AG in Figure 3.

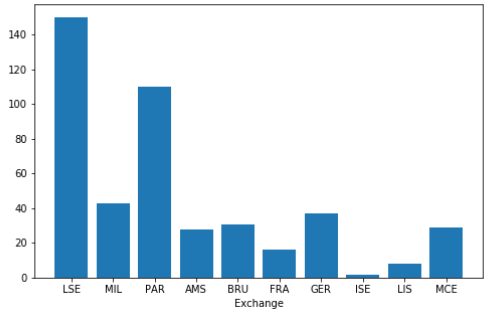

Figure 2: Stock distribution on exchanges

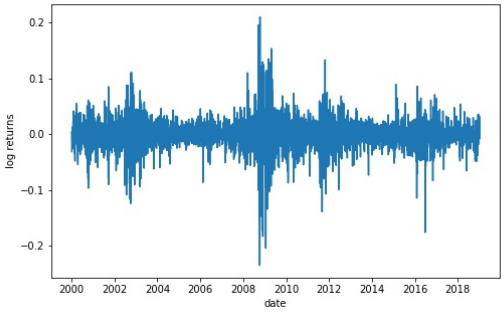

Figure 3: Log-return of Credit Suisse Group AG

3 Mean-variance and Minimum-variance Analysis Under Covariance Matrix

3.1 Selection of covariance matrix

All the covariance we used following are either normal covariance matrix(no correlation cleaning) or eigenvalues clipping covariance which keeps the top eigenvalues and shrink the others to a constant T.

We did both in-sample experiment and out-of-sample experiment. For in-sample experiment, we use all the data to find the weights of stocks as our portfolio. The covariance matrix is calculate by 4980 days and 412 stocks. The size of the covariance matrix is 412×412. For out-of-sample experiment, we choose 2000 days and 412 stocks. The covarince matrix also has the same size. We applied rolling window method in which way we computed portfolio based on previous data to predict next several days expect returns and then updated the previous data to compute new portfolio.

3.2 Mean-variance ananlysis

Mean-variance analysis is a simple method but widely used in finance field, which was first introduced by Markowitz [1]. Under the assumption that portfolio choice only depends on the mean and variance of end-of-period wealth. Mean-variance efficient portfolio has the lowest variance for a given expected return and highest expected return for a given variance.

The mathematics expression is as following:

$$\max_{\omega}(\omega'\mu - \frac{a}{2}\omega'\Sigma\omega) \qquad s.t \;\; \omega'\mathbb{1} = 1 \tag{1}$$

where μ is the vector of asset expected returns, Σ is the covariance matrix of returns, and ω the vector of portfolio weights.

We used all the data to develop the strategy of portfolio and show the performance of this strategy in Figure 4.

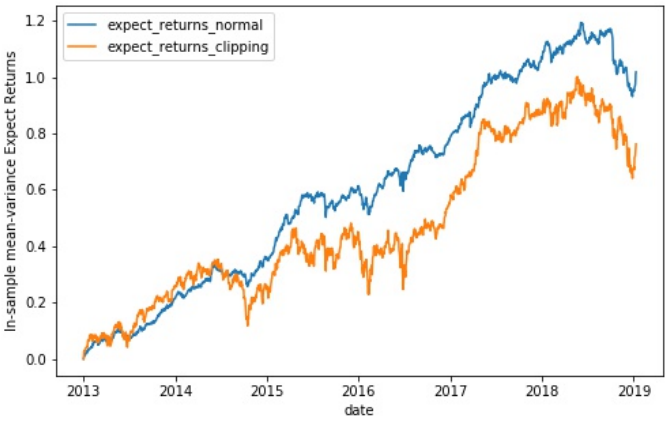

Figure 4: The in-sample performance of Mean-variance strategy with different covariance matrix.

3.3 Minimum-variance analysis

The minimum variance portfolio is a portfolio that minimize the volatility. The minimum-variance portfolio with expected return μ_p is the solution ω to

$$\min_{\omega} \frac{1}{2}\omega'\Sigma\omega \qquad s.t \;\; \omega'\mathbb{1} = 1, \;\; \mu'\omega = \mu_p \tag{2}$$

Where μ, Σ, ω represent the same thing as above. We also use all the data to develop the in-sample strategy of the portfolio. The performance is shown in Figure 5.

3.4 Rolling window and result presentation

For out-of-sample experiment, we decided to use rolling window to update our strategy. We choose window = 2000 days to develop our strategy on both mean-variance and minimum-variance analysis. Because of the computational limit, we decided to update per 30 days. For example, we use returns from day 1 to day 2000 to calculate the weight. Then we construct our portfolio using this weight from day 2001 to day 2030. After that we use returns from day 31 to

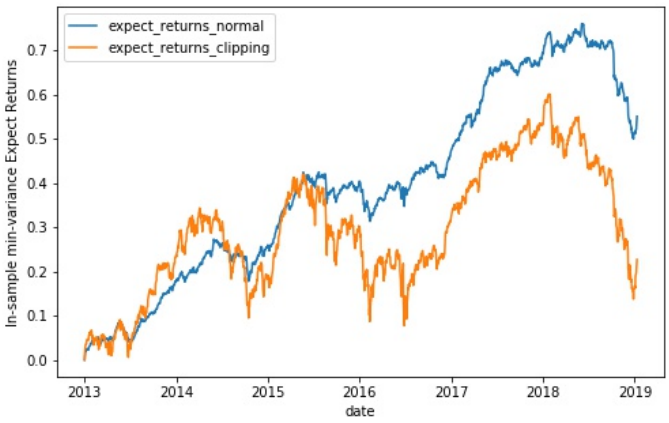

Figure 5: The in-sample performance of Minimum-variance strategy with different covariance matrix.

day 2030 to update our weight and use it to the following 30 trading days. Figure 6 is the out-of-sample performance of mean-variance and minimum-variance strategy with no correlation changing covariance matrix and eigenvalues clipping covariance matrix.

Figure 6: The performance of portfolio under Mean-variance and Minimum-variance strategy with different covariance matrix.

3.5 Conclusion

As we can see from above, the in-sample performance is better than out-of-sample performance under both covariance matrix and strategies. When we just look at the out-of-sample performance, i.e. the rolling window srategy, we can see that mean variance strategy performs better, which is in line with the theoretical knowledge. Comparing the performance of two

covariance matrix, we find that no correlation cleaning matrix has a better performance than clipping matrix.

4 Prediction with LSTM

4.1 LSTM

The Long Short-Term Memory network [2]–usually just called LSTM–is a type of recurrent neural network capable of modeling sequence, e.g. music, time series, etc. and learning long-term dependencies. A typical unit of LSTM is shown in Figure 7. When training with a sequence of data, each unit has the ability to learn to "memorize" important information of previous units and "forget" less important information.

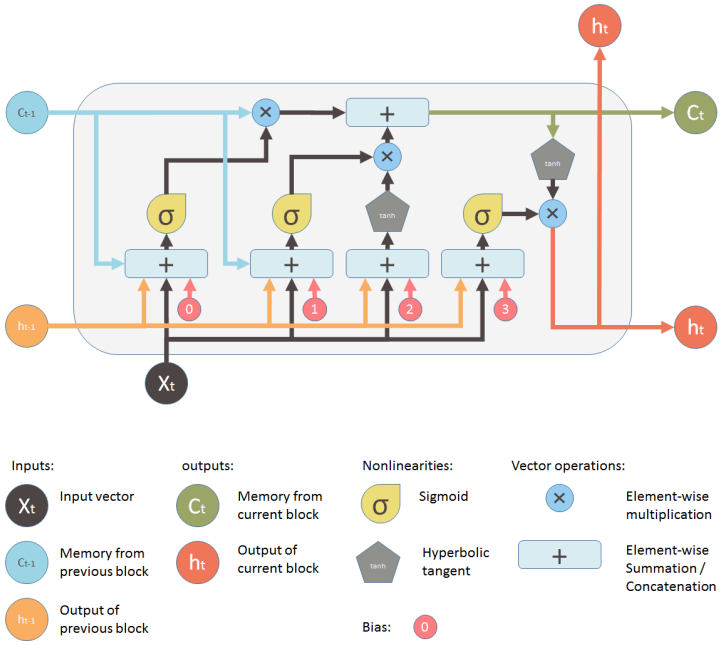

Figure 7: LSTM structure (Source)

We utilized a two-layer stacked LSTM with dropout layers [3] to avoid overfitting (which means the model too closely fits to a limited set of data points). The model is implemented using Keras [4].

Our aim is to predict closing price of one stock using all information (opening, highest, lowest, and closing price) from previous days, which is 60 days in our project. Among all the stocks we have, we choose Credit Suisse and Novartis, two swiss company to test our model's performance.

4.2 Data preprocessing

We first performed log-transformation to normalize our data (all features and targets) into a smaller range. Then, we built our training set with data before 2016 and the rest is our test set. After that, we split our data into samples of sequences of length 60 for LSTM input requirement.

4.3 Training setting

We use Mean Square Error as training metric and Adam [5] as optimizer. For each stock, the model is trained for 10 epoches in batch size of 64.

4.4 Prediction result

We trained our model on training set and run prediction on both training and test set. As shown in Figure 8 for Credit Suisse and Figure 9 for Novartis, the prediction results of training set (in orange) and test set (in green) are very close to our true closing price (in blue).

As model is trained on training set, it's natural to have good prediction results on training set, which means our model fits the training data very well. For test set, we can also achieve very good performance which shows the power of LSTM. Quantitatively speaking, in log scale, we have test set RMSE of 0.183 for Credit Suisse and of 0.040 for Novartis.

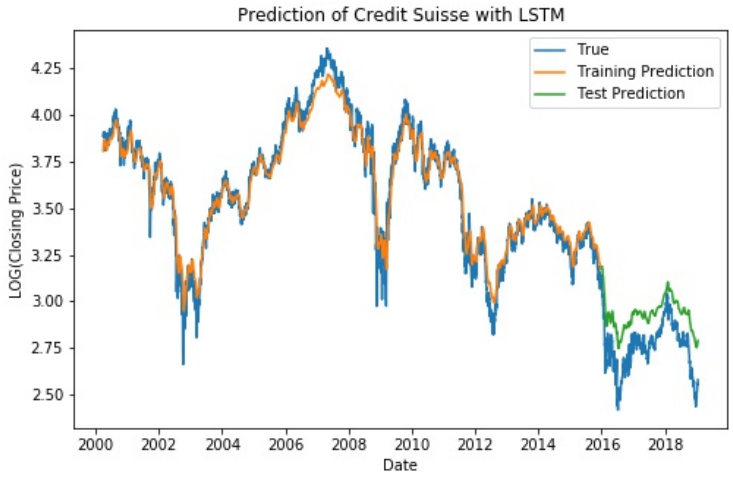

Figure 8: Prediction results of Credit Suisse using LSTM.

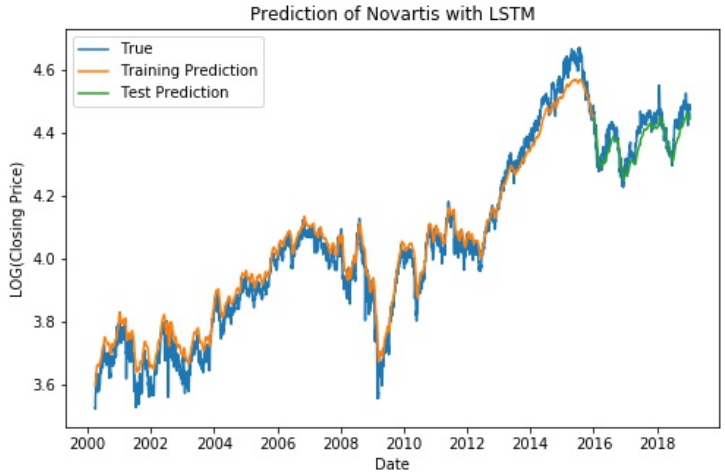

Figure 9: Prediction results of Novartis using LSTM.

5 Discussions

5.1 Challenges

There are several challenges we met during our project. The biggest one was computing the weights of portfolio based on rolling window method. The minimization function we used cost too much time for computing weights; we need to update data inside window and compute the weights again in each loop. We tried to use python library `Dask` to do parallel computation, but there exist several bugs because of the conflicts between the new version of `Dask` and other libraries. To address this problem, we selected proper window size and threshold of NaN values percentage to reduce the time cost (However, even with this method, we had to run the code for the whole night).

Besides, we tried to apply `shrinkage` covariance matrix when using mean-variance and minimum-variance analysis. However, we had problem using `sklearn.covariance.LedoitWolf` and could not move forward.

5.2 Improvement

For the analysis under covariance matrix, we can also try shrinkage method or Rnyi entropy-optimal portfolios. We could also choose and compare different window size and frequency of changing strategy.

For LSTM model, it shows some power in modeling the stock data, however, we didn't fit a model for each stock and it might not be universally good. Actually, we tried stocks like Google and Amazon, and it performed less satisfyingly than what's shown above. Moreover, the model used in our project is simple and straight-forward, thus, more advanced models or combined models can be applied to improve prediction performance. There are a lot of powerful models out there, such as gated recurrent units (GRUs), or classical machine learning methods (linear regression, support vector machine, etc.).

References

[1] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[4] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980