2017 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2017, Cancun, Mexico, December 4-7, 2017

Predictability of Financial Market Indexes by Deep Neural Network

Tomoya Onizawa, Takehiro Suzuki, and Tomoya Suzuki

Graduate School of Science and Engineering, Ibaraki University,
Nakanarusawa-cho 4-12-1, Hitachi-shi, Ibaraki, 316-8511, Japan.
Email: 17nm909h@vc.ibaraki.ac.jp, 17nm926l@vc.ibaraki.ac.jp, tomoya.suzuki.lab@vc.ibaraki.ac.jp

Abstract—We investigated the predictability of TOPIX (Tokyo Stock Price Index) as a typical stock market index by using various deep neural networks to learn nonlinear dynamics hidden in complex price movements. If there are not any dynamics in them, that is, financial systems are random, prediction accuracy would not be improved by deep learning procedures. However, we confirmed this improvement by some simulations with real stock data, which means that financial systems are not random and have some predictability.

1. Introduction

As a core concept of traditional financial economics, the efficient market hypothesis (EMH) insists that financial price behavior is completely unpredictable and can be characterized by the random walk. However, the EMH is merely a hypothesis, and especially the behavioral economics has insisted that the EMH is limited due to the psychological bias of market participants. Moreover, in recent years, it becomes easier to obtain real financial data, and therefore it is possible to discuss the validity of the EMH by using real data.

Because the EMH insists that financial markets are unpredictable, if we could show the predictability, it would be a counter-evidence of the EMH. In this purpose, we can examine the predictability of individual stocks, but the market indexes such as TOPIX and the Nikkei Stock Average (Nikkei 225) are more universal as market behavior. In particular, TOPIX is composed by the market capitalization of all the companies listed on the first section of the Tokyo Stock Exchange (TSE). Therefore, the present study focuses on TOPIX as a universal example to discuss the predictability of stock markets and the validity of the EMH.

As a nonlinear prediction model, we use the neural network framework. Although we can apply the time-series data of TOPIX to a neural network, it might be better to apply all the stocks composing TOPIX as explanatory variables because TOPIX depends on them. However, because the dimension of the input layer increases very much, the usual three-layer neural network might be hard to work in terms of the curse of dimensionality. For this reason, we need to use multilayered neural networks, but the vanishing gradient problem might be happen in the back propagation algorithm. To moderate this problem, we also use the deep neural network (DNN), which includes the pre-training by the autoencoder[1]. In addition, the dropout technique[2] can be also included in the DNN by suspending some neurons randomly to restrain the over-fitting problem.

If the EMH is true, the above attempts must be useless. However, if the prediction accuracy of TOPIX is improved, this means that stock markets have some patterns against the random walk and have some predictability against the EHM. To verify this possibility, we perform some predictive simulations with real stock data.

2. Traditional Neural Networks

2.1. Three-Layer Neural Network (3NN)

The most typical neural network is composed of the three layers: an input layer, a hidden layer, and an output layer. First, let us denote the output value from the j-th hidden layer as

$$y_j = f\left(\sum_{i=1}^{I} w_{ij}x_i + b_j\right), \quad \text{where} \quad f(u) = max(u, 0). \tag{1}$$

Here, x_i is the output value from the i-th input layer, w_{ij} is the connection weight from the input layer to the hidden layer, b_j is the bias of the hidden layer, and f is a nonlinear function. Then, $i = 1 \sim I$ and $j = 1 \sim J$.

Similarly, the output value from the k-th output layer is denoted as

$$z_k = g\left(\sum_{j=1}^{J} w_{jk}y_j + b_k\right), \quad \text{where} \quad g(u) = u. \tag{2}$$

Here, w_{jk} is the connection weight from the hidden layer to the output layer, b_k is the bias of the output layer, and g is the identity function. Then, $k = 1 \sim K$.

Next, the model parameters: w_{ij}, w_{jk}, b_j, and b_k are trained by the back propagation algorithm. Moreover, for

dynamically adjustment of the training late, we apply the adaptive moment estimation (Adam) [3], which estimates the mean $m(t)$ and variance $v(t)$ of the distribution of the error function $E(t)$,$(t = 1, 2, \cdots, T)$. Here, $E(t)$ is the square error between a predicted value and its true value at time t.

The update equation of training late is as follows:

$$m(t+1) = \beta_1 m(t) + (1-\beta_1)\nabla E(t)^2, \quad (3)$$

$$v(t+1) = \beta_2 v(t) + (1-\beta_2)\nabla E(t)^2. \quad (4)$$

Next, these values are modified as follows:

$$\hat{m}(t+1) \leftarrow m(t+1)/(1-\beta_1), \quad (5)$$

$$\hat{v}(t+1) \leftarrow v(t+1)/(1-\beta_2). \quad (6)$$

Here, β_1 and β_2 is used to prevent the rapid decline of training late. Moreover, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are recommended in the original study[3], and therefore we adopt the same values.

Finally, each model parameter of a neural network is renewed by the back propagation with the above values.

$$w_{jk}(t+1) = w_{jk}(t) - \alpha\hat{m}(t+1)/(\sqrt{\hat{v}(t+1)} + \epsilon) \quad (7)$$

$$w_{ij}(t+1) = w_{ij}(t) - \alpha\hat{m}(t+1)/(\sqrt{\hat{v}(t+1)} + \epsilon) \quad (8)$$

$$b_k(t+1) = b_k(t) - \alpha\hat{m}(t+1)/(\sqrt{\hat{v}(t+1)} + \epsilon) \quad (9)$$

$$b_j(t+1) = b_j(t) - \alpha\hat{m}(t+1)/(\sqrt{\hat{v}(t+1)} + \epsilon) \quad (10)$$

Here, $\alpha = 0.001$ and $\epsilon = 10^{-8}$ are recommended in Ref.[3], and therefore we adopt the same values.

2.2. Deep Neural Network (DNN)

As the number of neurons in the input layer is larger, the relationship between input and output data becomes more complex, and therefore we need a multilayered neural network (MNN) to enhance its learning ability. Moreover, in terms of the curse of dimensionality, the MNN is useful for dimension reduction, which is a sort of feature selection. However, in the back propagation mentioned above, the MNN has the vanishing gradient problem due to which the modification of model parameters becomes weaker as in closer to the input layer.

However, recently, the deep learning was invented, and one of core concepts is the autoencoder[1] that can be used for the pre-training of MNN to moderate the vanishing gradient problem. Here, this deep learning model is called the deep neural network (DNN), and we also apply this model to examine whether the predictability of TOPIX can be improved.

Moreover, the dropout[2] is also a core concept of the deep learning in order to reduce the over-fitting problem because the DNN tends to be complex. Then, its benefit can be explained from the viewpoint of the ensemble learning. However, if financial systems are completely random, these concepts must not work at all.

3. Stacked Autoencoder

Some autoencoders are stacked for the pre-training of DNN as shown in Fig. 1. An autoencoder is just a 3NN of Eqs. (1) and (2), and its model parameters can be learned by the back propagation mentioned in Sect. 2.1. However, the number of neurons in the hidden layer is less than that in the input layer, and the number of neurons in the output layer is the same as the input layer. Namely, if we can get the same input data from the output layer, it can be said that the hidden layer worked for dimension reduction and the encoded input data can be decoded without losing essential information.

If we composed the DNN that has more than four layers, we stack the autoencoder as shown in Fig. 1 for pre-training of the DNN. First, we remove the output layer of the first autoencoder, and we consider the hidden layer of it as the input layer of the second autoencoder. Next, we train the second autoencoder by the back-propagation algorithm, and repeat the same procedure many times. Finally, the output layer is connected to the hidden layer of the final autoencoder where the original input data were compressed into a low dimension by the stacked auto encoders. Only the output layer is initialized at random. In our study, the number of neurons in each hidden layer was reduced by 90% compared to each input layer. After the pre-training, the whole DNN is trained again by the back-propagation algorithm. Although the vanishing gradient problem happens again in the pre-trained DNN, the whole network even near the input layer was already trained. Therefore, this problem is not serious.

4. Predictive Simulations

In this study, we predict the return rate $y(t)$ of the TOPIX futures during the daytime from the open (9:00) to the close (15:00) by using the return rates of stocks $x_i(t)$, $(i = 1, 2, \cdots, I)$ during the night time from the close to the open. Namely, $y(t)$ is the output of a neural network and the set of $x_i(t)$ is the input of it:

$$y(t) = \frac{c(t) - o(t)}{o(t)}, \quad (11)$$

$$x_i(t) = \frac{o_i(t) - c_i(t-1)}{c_i(t-1)}, \quad (12)$$

where $c(t)$ and $o(t)$ are the closing and opening price of the TOPIX futures, and $o_i(t)$ and $c_i(t-1)$ are the opening and closing price of the i-th stock. Then, each stock was selected from the first section of Tokyo Stock Exchange (TSE) and some stocks having missing data were omitted. The total number of the used stocks I was 825. All real data were obtained in the Yahoo Finance[4]. Figure 2 shows the

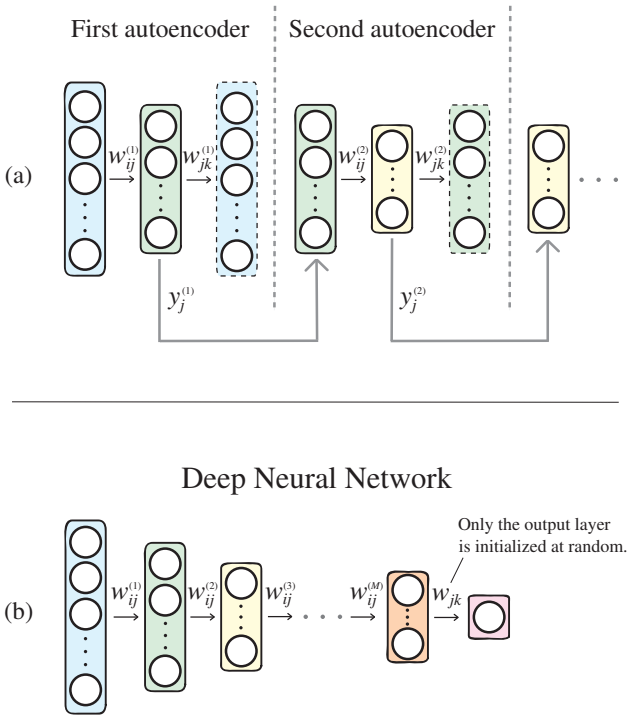

Figure 1: Concept of the stacked autoencoder

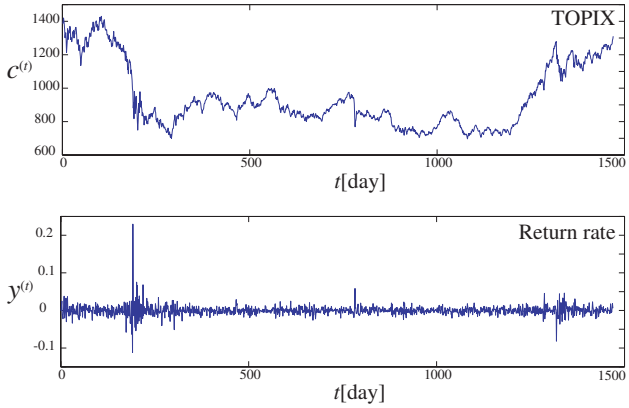

Figure 2: Closing price $c(t)$ of the TOPIX futures and its return rate $y(t)$ during 2008–2013

closing price $c(t)$ of the TOPIX futures and its return rate $y(t)$ during 2008–2013.

The mode parameters of 3NN and DNN were trained by $y(t)$ and $x_i(t)$ during 2008–2012, and then these trained models were tested during 2013. As a prediction accuracy, we evaluated the correlation coefficient R:

$$R = \frac{\sum_{t=1}^{T}(\hat{y}(t) - \langle\hat{y}\rangle)(y(t) - \langle y\rangle)}{\sqrt{\sum_{t=1}^{T}(\hat{y}(t) - \langle\hat{y}\rangle)^2}\sqrt{\sum_{t=1}^{T}(y(t) - \langle y\rangle)^2}}, \qquad (13)$$

where $\hat{y}(t)$ is a predicted value during the test period, $y(t)$ is its true value, and $\langle\cdot\rangle$ means the average value.

Then, we examined the efficiency of the multilayered network, the pre-training by the stacked autoencoder, and the dropout. If financial systems are completely random, these techniques are useless. Namely, we compared the following six cases:

(a) 3NN without pre-training or dropout

(b) 3NN with dropout

(c) DNN without pre-training or dropout

(d) DNN with dropout

(e) DNN with pre-training

(f) DNN with pre-training and dropout

Here, because the 3NN cannot apply the stacked autoencoder, its pre-training was not considered.

Figure 3 shows the prediction accuracies R given by the neural networks (a)–(f) during the test period. First of all, the three-layer neural network (3NN) did not work even with the dropout due to the curse of dimensionality. For this predictive simulation, the dimension of the input layer is 825. It might be too many for the 3NN. On the other hand, the deep neural network (DNN) worked better than the 3NN. Especially, we can see the difference among (c)–(f) and the number of hidden layers M. In the case of the DNN without the pre-training: (c) and (d), $M = 3$ shows the best performance. However, in the case of using the pre-training: (e) and (f), the larger M shows the best score. This means that multilayered networks can reduce the high-dimensional problem. However, if the multilayered network does not apply the pre-training, its performance cannot be improved due to the vanishing gradient problem. In addition, as the number of hidden layers M is larger, the decoding error by the autoencoder becomes larger and namely the pre-training does not work well. For this reason, the performance of each DNN has a peak according to M.

Next, we investigated the improvement rate α of each DNN realized by the dropout and/or the pre-training. For example, the improvement rate from the case (c) to (d) was calculated by

$$\alpha(c, d) = \frac{R(d) - R(c)}{|R(c)|}, \qquad (14)$$

where $R(d)$ is the prediction accuracy given by the case (d). Figure 4 shows the results. We can confirm the improvement given by the dropout and the pre-training. Namely, these techniques worked well as we expected even in financial markets. This means that financial markets have

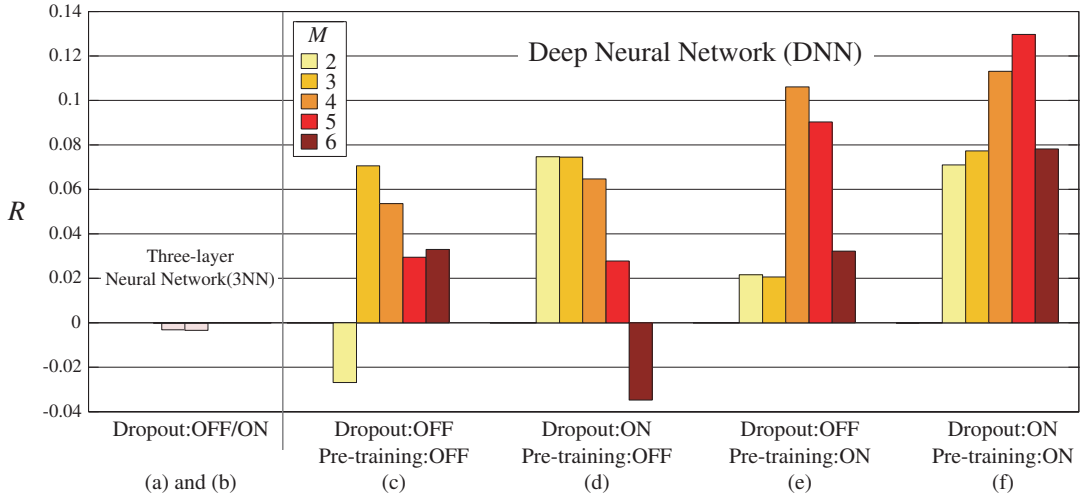

Figure 3: Prediction accuracies R given by the neural networks (a)–(f) during the test period. M is the number of hidden layers in a deep neural network.

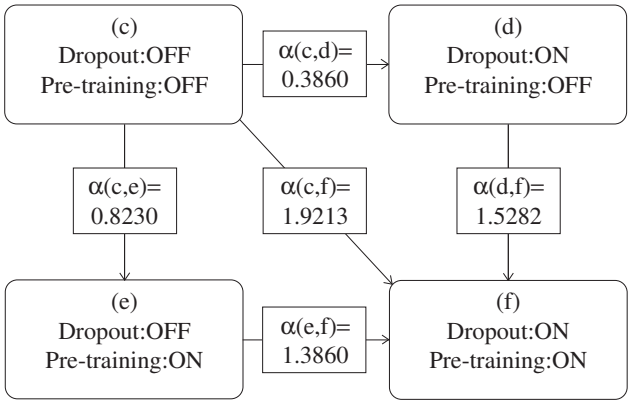

Figure 4: The improvement rate α between two cases. Each α was calculated by the mean value of M=2–6

some patterns differently from the random walk, and this is a counter-evidence of the EMH.

5. Conclusion

We investigated the predictability of TOPIX futures as a general example of financial markets by using many individual stocks as explanatory variables. For this reason, we applied the deep neural network to avoid the curse of dimensionality, and also adopted the pre-training by the stacked autoencoder to moderate the vanishing gradient problem and the dropout to reduce the over-fitting problem.

As results, we confirmed that multilayered networks can improve the prediction accuracy of the TOPIX if the pre-training is applied. Moreover, the efficiency of the dropout was also confirmed as we expected. These results conclude that financial markets have some patterns differently from the random walk. That is why more advance prediction models could learn these patterns better and improved the prediction accuracy. This improvement corresponds to a counter-evidence of the EMH.

As future problems, in order to enhance the universality, we need to increase the number of trials with longer test periods, and apply the surrogate data test to compare with random phenomenon.

References

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle: "Greedy layer-wise training of deep networks," *Proc. NIPS*, Vol.19, pp153–160, 2006.

[2] N. Srivastava, G. E. 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.

[3] J. Ba and D. P. Kingma: "Adam: A method for stochastic optimization," *Proceedings of International Conference on Learning Representations*, 2015.

[4] Yahoo! Finance: http://stocks.finance.yahoo.co.jp