Journal of Computational Information Systems 10: 2 (2014) 883–893
Available at http://www.Jofcis.com

A Stock Decision Support System based on DBNs [⋆]

Chengzhang ZHU [1,∗], Jianping YIN [2], Qian LI [3]

[1] *College of Computer, National University of Defense Technology, Changsha 410073, China*

[2] *State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China*

[3] *School of Economics, Minzu University of China, Beijing 100081, China*

Abstract

Stock trading is a common way to obtain benefits. Some trading decision support systems have played important roles in people's trading. However, most of them cannot guarantee a good rate of return because of the uncertainty and noise in stock market. It would be ideal if a rough prediction of stock price and a robust trading strategy could be combined. In this paper we report our attempt towards making trading decision using oscillation box theory based on deep belief networks (DBNs). First of all, the DBNs are used to in-depth learn historical data of stock transactions and roughly forecast the stock price time-series in the future period. Meanwhile, the gray relation degree, to obtain a more accurate result, is used to determine the weight of the factors affecting the stock market. Based on it, the oscillation box theory is introduced to make decision, which can eliminate the influence of local low precision. We implement an automatic stock trading decision support system to assists people in decision making on stock trading by suggesting users buying or selling stock. In experiment, we tested trades on 4 typical stock movements and used 400 stocks in S & P 500 to examine the performance of our system. According to the experiments the historical data learning of stock transactions costs a lot of time, the system is fully capable of providing suggestions on stock trading decision-making.

Keywords: Stock Predict; Deep Learning; DBNs; Oscillation Box Theory

1 Introduction

In order to get a fruitful profit, people often tend to find a reliable way to predict the stock and make lucrative investment decisions. However, the stock time-series forecasting is regarded as one of the most challenging applications of time-series forecasting since stock market indices are essentially dynamic, nonlinear, complicated, nonparametric and chaotic [1]. There has been a lot of work trying to forecast the stock prices but nobody can predict a stock market with high enough accuracy. However, one could be able to predict the price's overall trend based on stock

[⋆]This work was supported by the National Natural Science Foundation of China (Project No. 60970034, 61105050, 61170287, 61232016).

[∗]Corresponding author.

Email address: kevin.zhu.china@gmail.com (Chengzhang ZHU).

1553–9105 / Copyright © 2014 Binary Information Press
DOI: 10.12733/jcis9653
January 15, 2014

transactions historical data. Therefore people can combine the result with some well trading strategy to obtain rich rewards. Obviously, people try to find a successful prediction method that can achieve best results using minimum required input data and the least complex stock market model [2]. But the traditional model cannot well comprehensively descripts the stock market since it is affected by many factors such as political events, firms policies, general economic conditions, investors expectations, institutional investors choices, movement of other stock market, and psychology of investors etc [3]. Lots of attentions have been devoted to applying different methods especially neural networks, which can obtain a comparatively accurate solution in the complex and noisy environment, into stock prediction [4, 5, 6]. Some of them have made breakthrough, which make the stock market prediction significantly accurate and robust. However, the accurate of all of them need further improve since subtle differences can lead to large fluctuations in gains.

In order to achieve high precision in prediction, we focus on the deep machine learning, which imitates the brain's cognitive process and has been widely used in extraction and recognition of the information characteristics [7, 8]. A method that under the deep machine learning architecture, which is named deep belief networks (DBNs), has been proved to have excellent performance [9]. It has the capability to address issues associated with applying back-propagation to deeply layered nueral networks traditionally, such as long learning time, necessity of a substantial labeled data set for training, and inadequate parameter selection techniques that lead to poor local optima [10]. For this reason, we can introduce DBNs to forecast the trends of stock price to get a higher accurate result.

At the same time, we draft trading strategies based on the oscillation box theory, which indicated that the stock price was always floating up and down in a certain range of a box in a period of time [11]. When the price is closing to the upper boundary of this box, it will fall in the future. But if it breaks the boundary, the price trend will entry another oscillation box and starts a whole new upward or downward trend. The theory is also suitable for the contrary side when stock price is closing to the lower boundary. In this case, the top priority for us is to identify the boundary of the box and find a way to confirm whether the price goes beyond the boundary or not.

In this paper, we implement an automatic stock decision support system, which utilizes DBNs to predict the highest and lowest stock price in the next period as the upper and lower boundary of the oscillation box and give an inspection rule in order to confirm whether a stock price goes beyond the boundary or not. Since it has the capacity of in-depth study the historical data, the system can accurately predict the boundaries of the box, which ensuring the transaction making large gains. According to experiments, our system has an obvious advantage when compared with the buy-and-hold strategy.

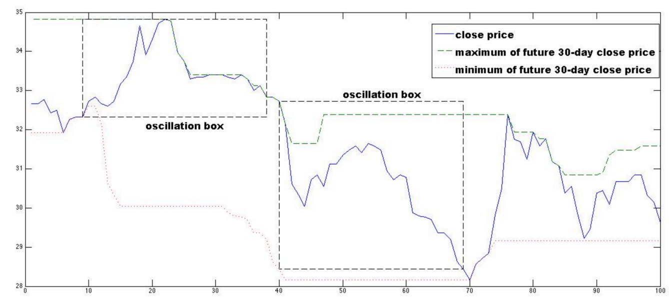

Fig. 1: The oscillation box

The remainder of this paper is organized into four sections. Section 2 briefly reviews the oscillation box theory and the deep belief networks theory. Section 3 provides an explicit explanation for our system's trading strategy. Section 4 shows the experiments with analysis, and ultimately. Section 5 contains the concluding remarks.

2 Related Work

2.1 Oscillation box theory

The oscillation box theory, which proposed in 2007 by Nicolas [11], believes that the stock price has its mobility scale in a period of time. Thus it has the maximum price ensures an upper boundary and the minimum price ensures a lower boundary. Theses two boundaries represented a frame, which is called the oscillation box. Once decided dimensions of the box, the stock could do what it liked, but only within that frame. And in this box, the stock price has a rising trend when it closes to the lower boundary and has a downing trend on the contrary. On the other hand, the price will go into another box to start a new trend in a range after it breaks through the boundary of this certain box. The oscillation box is showed in Fig. 1. We realize that if we buy the stock when the price breaks the upper boundary and sell it as soon as it breaks the lower boundary, our buying, based purely on the stock price box theory, enable us to get a fruitful profit without knowing anything else about it.

However, the problem is: How to detect this change? How to judge a movement at the time it happens? In our system we proposed a method to detect it automatically based on the deep belief networks, which is far more reliable than people's past experience.

2.2 Deep belief networks

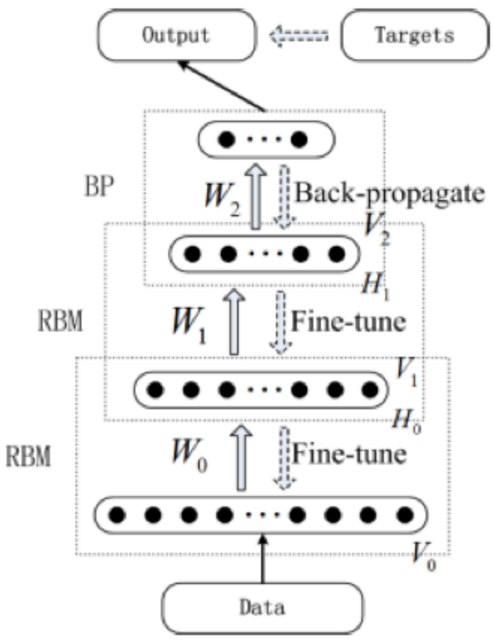

Fig. 2: The framework of DBNs

DBNs, initially introduced in [9], are multilayer probabilistic generative models, where each layer encodes statistical dependencies among the units in the layer below. It often consists of several restricted Boltzmann machine (RBM) layers and a BP layer. The framework of the DBNs is illustrate as Fig. 2.

The RBM includes a single visible layer and a single hidden layer. If we defined the set of visible units as $\vec{v}$ and the set of hidden units as $\vec{h}$, the probabilistic semantics for a RBM can be presented as follow:

$$P(\vec{v},\vec{h}) = \frac{1}{Z} exp(-E(\vec{v},\vec{h})), \tag{1}$$

where Z is the partition function, which is given by:

$$Z = \sum_{\vec{h},\vec{v}} exp(-E(\vec{v},\vec{h})), \tag{2}$$

and E is the energy of the joint configuration $(\vec{v},\vec{h})$, which is given by:

$$E(\vec{v},\vec{h}) = -\sum_{i,j} v_i W_{ij} h_j - \sum_j b_j h_j - \sum_i c_i v_i, \tag{3}$$

where W_{ij} represents the symmetric interaction term between visible unit i and hidden unit j while b_j and a_j are their bias terms. From functions above, it is clear that the hidden units are conditionally independent of one another given visible layer, and vice versa. Therefore, the efficient block Gibbs sampling can be performed by alternately sampling each layer's units given the other layer. The unit's expected value is often referred as its activation. Although the RBM is always limited in what it can represent by itself, its real power emerges when many RBM layers are combined to form a deep belief network. The DBNs greedily trains each layer from lowest to highest, while each RBM layer independently learns its parameters and the previous layer's activations are used as the inputs of the next layer. In this process, RBM keep the two adjacent layer's information as same as possible. All the learning above is unsupervised. On the top layer of DBNs is always a supervised BP layer, which fine-tunes the whole model and obtains the output results.

After multiple RBM layers, the wrong or insignificant information is weakening and eliminating layer by layer, and finally become more nature and abstract information. Therefore, the higher layer it is more representation power of the whole model it has. The final data used by BP layer consist of sophisticated features, which reflect the structured information, promote a better performance than direct original data.

2.3 Gray correlation degree (GCD)

The gray correlation degree, which is proposed by Deng in [12], is introduced to present the relational degree between two data sequences by using the geometric shape of sequence curves. It turns out that the closer the two curves are, the higher degree is it. The main idea of the method is to use a sequence showing the geometry of the curve between two data sequences correlation. The closer the two curves are, the higher degree it gives to us [13]. If we have a feature $\vec{X} = [x_1, x_2, \cdots, x_n]$ and target $\vec{T} = [t_1, t_2, \cdots, t_n]$ we can calculate the feature's GCD as follow:

$$r(t_i, x_i) = \frac{min|t_i - x_i| + \xi max|t_i - x_i|}{|t_i - x_i| + \xi max|t_i - x_i|} \tag{4}$$

$$r(\vec{T},\vec{X}) = \frac{1}{n}\sum_{i=1}^{n} r(t_i,x_i), \tag{5}$$

where $\xi \in (0,1)$ is the discernibly coefficient which often set to 0.5. The $r(t_i,x_i)$ is the gray correlation degree of $\vec{T}$ and $\vec{X}$ at ith point. The $r(\vec{T},\vec{X})$ is the gray correlation degree of $\vec{T}$ and $\vec{X}$.

3 Detail of the Decision Support System

The decision support system includes the following steps. In first step, it scales related indicators, which are calculated from the historical stock transaction data. Then it extracts relationships between scaled indictors and stock price time-series. These relationships' degree will be set as input values weight of the DBNs. Next step comes to train the DBNs, while weighted indictors sequence as input vectors and stock history price time-series as target vectors. The third step is predicting the stock price sequence for the next period of time in the future by this trained DBNs. Thus, the system can get the lower and upper bounds, which will be set as the minimum and maximum values of stock prices respectively, of the oscillation box. Finally, the system is able to decide whether transact or not in accordance with the trading strategy based on oscillation box theory. The architecture of the system is showed as Fig. 3.

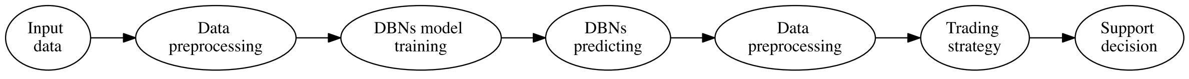

Fig. 3: Support system architecture

3.1 Pre-processing history indictors

The system's target value is the daily stock price, which is represented by the daily stock closing price. Some indicators are selected as the DBNs' input feature values, such as OPEN, HIGH, LOW, CLOSE, VOL, AMOUNT, MA, ROC, RSI, FASTK, SLOWK, SLOWD. The computations can be found in [14]. The boundary in a period of days can be used as features in order to detect the boundary in next few days through prediction the stock price time-series. In this system, we use the highest price and lowest price of the stock in the future n_1 days as the upper boundary Up_k and lower boundary Low_k. $Up_k = max(C_{i+1}, C_{i+2}, \cdots, C_{i+n_1})$, $Low_k = min(C_{i+1}, C_{i+2}, \cdots, C_{i+n_1})$ were C_k represent the closing price in the kth day.

There are totally 14 indictors and 1 target in our system. If a indictor data sequences are $X = (x(1), x(2), ..., x(n))$, all data of the indictor will be normalized to $[-1, 1]$ by

$$x(i)_{normalize} = -1 + 2\frac{x(i) - min(X)}{max(X) - min(X)} \tag{6}$$

Otherwise, the prediction result will be denormalized by

$$p_{denormalize} = \frac{p(max(X) - min(X)) + max(X) + min(X)}{2}, \tag{7}$$

where p is the prediction result.

However, different indicators have quite different influence degrees when predicting the stock price. In this case, the indicators, which have a relatively large influence, are enlarged in the prediction so that we can have a more accurate prediction result. Thus we use the gray relation analysis method to get relationships of the scaled indicators and stock price time-series and use it as the input weights w_i.

3.2 Stock prices prediction based on DBNs

In order to identify the box boundaries, we use DBNs to learn history stock data in n_2 previous days and then predict prices in next n_1 days. The input vector for DBNs learning can be defined as follow:

$$\vec{I_i} = \vec{F_i} \circ \vec{W_i}, \tag{8}$$

where $\circ$ is Hadamard product, i presents ith day. $\vec{F_i}$ means the feature vector, which is defined as $F_i = [O_k, H_k, L_k, C_k, VOL_k, MA_k, ROC_k, RSI_k, FastK_k, SlowK_k, SlowD_k, Up_k, Low_k]$. And $\vec{W_i}$ is input weights vector, where w is the weight obtained by the GCD. The above mentioned k represents $k = (i-1, i-2, \cdots, i-n_2)$ and all features are indictors mentioned in Section 3.1. Meanwhile, the target vector for DBNs can be defined as $\vec{T_i} = [C_{i+1}, C_{i+2}, \cdots, C_{i+n_1}]$.

We chose data in n days before the testing data to train the DBNs. If we want to predict stock prices after ith-day, the input vectors can form a matrix as $\vec{I} = [\vec{I}_{i-n}, \vec{I}_{i-n+1}, \cdots, \vec{I}_{i-n_1}]$ while the target vectors can form a matrix as $\vec{T} = [\vec{T}_{i-n}, \vec{T}_{i-n+1}, \cdots, \vec{T}_{i-n_1}]$. We use Levenberg-Marquardt (LM), a combined quasi-Newton and gradient ascent algorithm, which was found to converge fastest and almost produce the best error, to train the top layer of DBNs. After training DBNs, which used $\vec{I}$ as input matrix and $\vec{T}$ as target matrix, we can predict stock price $\vec{T_i}$ using input vector $\vec{I_i}$. The upper boundary and lower boundary can be set as maximize and minimize price of $\vec{T_i}$.

3.3 Trading strategy

The trading strategy of our system is to buy stock when it's price break the upper boundary of the oscillation box and sell it on the contrary. Only if it satisfy two conditions below, the price can break the upper boundary. One is it must very close to the lower boundary of the new box. The other one is the lower boundary of the new box will move upward. Conditions are quite similarly when it breaks the lower boundary. Algorithm 1 detailed display of our trading strategy.

4 Experiments

We conducted a series of experiments to detect the availability and accuracy of our system. We chose 400 stocks in S & P 500 to test. These test samples, including the common trend of the stock form, such as the bull market, bear market and fluctuation market, are used for a comprehensive testing of the system performance. Meanwhile, we also discussed and tested the choice of the system parameters. All experiments were carried out in the MATLAB platform.

4.1 Performance evaluation

Two performance indicators is used in our experiments. One of them is MSE (mean squared error), which is used to illustrate the accuracy of DBNs regression. The MSE is defined as follow:

$$MES = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^*)^2, \tag{9}$$

where y_i is the actual output and y_i^* is the estimate.

The other is rate of profit which can defined as

$$\text{rate of profit} = (Y - Y_0)/Y_0 \times 100\%, \tag{10}$$

where Y is the money after trade and Y_0 is the initial money.

In our experiments, we suppose \$10000 initial money and use all money or stock to trade at each operation. As the real trading, we set the transaction cost of each trading 0.5 percent. Then we let the system trade on a stock for a period of time and get the average MSE and rate of profit in the final. Particularly, we short-selling of all stocks hold at the last day of test trading.

4.2 Trading of typical stock movements

A movement of fluctuant and bull market is showed as Fig. 4. The transaction rate σ is set to 0.02 and stop-loss rate θ is set 0.04 and φ, ϕ set to $0, 0.1$ respectively. In the experiment, our system can profit to 78.73% while the market gains about 37.91%. The MSE of the DBNs is 0.1306. The testing set is selected from ARG about 400 days.

A movement of fluctuant and bear market is showed as Fig. 5. The transaction rate σ is set to 0.01 and stop-loss rate θ is set 0.04 and φ, ϕ set to $0, 0.08$ respectively. In the experiment,

Algorithm 1 Trading strategy.

```
if next trade == buy then
  if $\frac{|C_i - L_i|}{C_i} \leqslant \sigma$ and $Low_i$ is in uptrend then
    if $sellprice - C_i \geqslant \varphi$ then
      Buy, $buyprice = C_i$
      next trade = sell
    end if
  end if
else if $\frac{|C_i - L_i|}{C_i} \leqslant \sigma$ and $Up_i$ is in downtrend then
  if $C_i - buyprice \geqslant \phi$ then
    Sell, $sellprice = C_i$
    next trade = buy
  end if
  if $\frac{buyprice - C_i}{buyprice}\theta$ then
    Sell, $sellprice = C_i$
    next trade = buy
  end if
end if
```

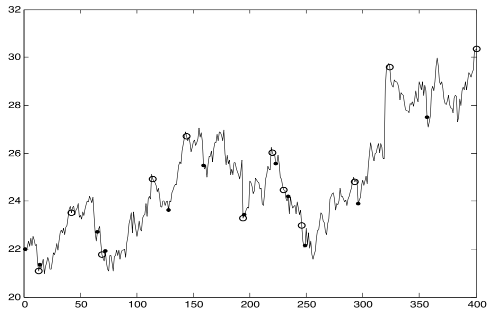
Fig. 4: The fluctuant and bull market movement, where • means buy point and ∘ means sell point

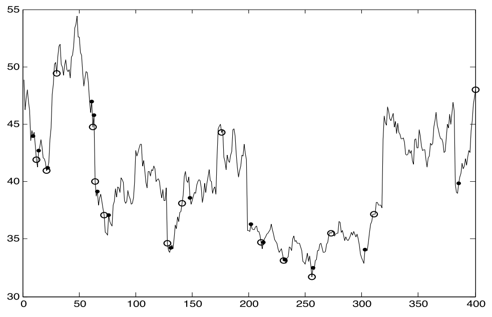
Fig. 5: The fluctuant and bear market movement, where • means buy point and ∘ means sell point

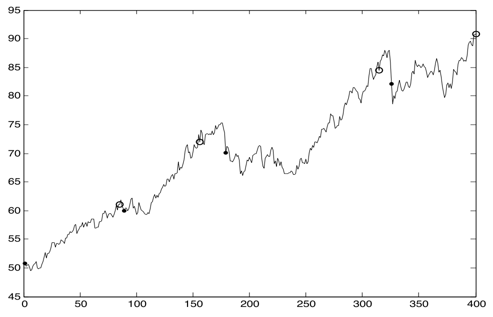
Fig. 6: The overall bull market movement, where • means buy point and ∘ means sell point

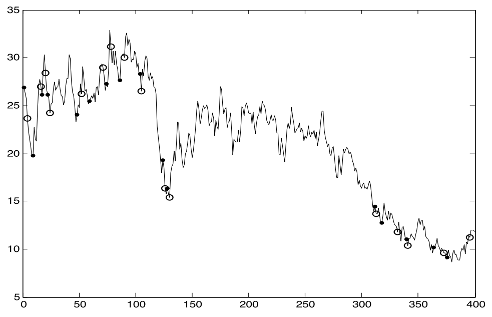
Fig. 7: The overall bear market movement, where • means buy point and ∘ means sell point

our system can profit to 21.40% while the market losses about 1.8%. The MSE of the DBNs is 0.1289. The testing set is selected from AMZN about 400 days.

A movement of overall bull market is showed as Fig. 6. The transaction rate σ is set to 0.03 and stop-loss rate θ is set 0.1 and φ, ϕ set to $0, 0.2$ respectively. In the experiment, our system can profit to 88.76% while the market gains about 79.00%. The MSE of the DBNs is 0.1246. The testing set is selected from AVB about 400 days.

A movement of overall bear market is showed as Fig. 7. The transaction rate σ is set to 0.01 and stop-loss rate θ is set 0.05 and φ, ϕ set to $0, 0.05$ respectively. In the experiment, our system can profit to 30.47% while the market losses about 56.01%. The MSE of the DBNs is 0.1252. The testing set is selected from ALTR about 400 days.

According to these results, our system significantly outperforms the buy-and-hold strategy in all typical stock movements.

4.3 Trade on S & P 500

We simulate trading of 400 stocks in 400 trading days, which are selected in S & P 500 from March 18, 2004 to October 17, 2005, to examine the average performance of the system. There are 224 of these stocks' movement are bull, the other are bear. In these trading test, the same parameters are used. The transaction rate σ is set to 0.01 and stop-loss rate θ is set to 0.05 and φ, ϕ set to 0,0.05 respectively. The test results are showed in Table 1. Clearly, in most cases our system is superior to the buy-and-hold strategy, and can gains a much higher average profit.

Table 1: Average performance of trading 400 stoks in S & P 500 for 400 days

Market pattern	Stock number	Less than buy-and-hold	Less(%)	loss Number	Loss(%)	Average profit(%)	Average profit of buy-and-hold(%)
bull	212	52	24.53	0	0	39.52	27.32
bear	188	6	3.19	20	10.10	10.32	-20.68
Total	400	58	14.50	20	5	**25.80**	**4.76**

4.4 The optimal sets of parameters

Firstly, the training data size n is condsidered. In the experiment above, the n is set as 1200 days, which is 3 times of testing data. However, the result will be different if this value changes because the DBNs will learn different knowledge. Usually, the more knowledge is learned the better result obtains. But there are some noise data in stock market. Thus a lager n may also lead to poor performance. On experience, a stock with bull movement always has less noise data over a longer period of time. Conversely, a stock with bear movement is often accompanied by more noise data. Therefore, we can control the n based on the stock trend. Now we test the same stocks as Table 1 used again with controlling n. All the other parameter are not change. Table 2 shows the result. The revenue at this time has been significantly increased.

Table 2: Average performance of trading 400 stock in S & P 500 for 400 days with window size control

Market pattern	Stock number	Less than buy-and-hold	Less(%)	loss Number	Loss(%)	Average profit(%)	Average profit of buy-and-hold(%)
bull	212	34	16.03	0	0	45.62	27.32
bear	188	0	0	8	4.04	12.33	-20.68
Total	400	34	8.50	12	3	**29.97**	**4.76**

Secondly, since we have used the previous n_2 days for feature value extraction and prediction, the extreme value stock may reach in next n_1 days, the n_1 and n_2 may be considered. We first fix n_1 as 15 and chance n_2 from 1 to 5 times of n_1 to find the optimal value of n_2. The result illustrates in Table 3, which shows that the most profit will be gained while n_2 is 4 times of n_1. Then we adjust n_1 from 5 to 20, and set n_2 as 4 times of n_1. At this time we can see the average yield arrive maximum when n_1 is set to 15 in Table 4.

Table 3: Average performance of trading 50 stock by varying n_2

	15/15	15/30	15/45	15/60	15/75
Average profit(%)	29.37	33.21	33.64	**34.24**	34.12
number of transaction	15.1	14.2	13.2	12.8	12.2

Table 4: Average performance of trading 50 stock by varying n_1

	5/20	8/32	10/40	15/60	20/80
Average profit(%)	27.96	31.57	32.34	**34.24**	33.32
number of transaction	14.1	13.3	12.3	12.8	12.0

Table 5: Performance of trading by varying σ

σ	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
Profit(%)	2.32	6.58	**8.22**	6.33	5.75	4.45	4.84	4.13	5.64	6.01
Number of transaction	8	9	9	9	9	9	9	9	9	10

The last but not the least, the σ also need to be considered. On one hand this parameter controls the speed of transactions, on the other hand it also eliminates some of the impact caused by the prediction error. For fluctuations or bull movement, we can set it to a larger value to get more transactions. In contrast, for the bear movement, it can be set a small value to reduce transactions. However, the parameter must be set within a range, otherwise it will reduce the accuracy. We selected a stock to show the impact for system profit of σ. The value of σ adjust from 0.005 to 0.04 while n is set to 1200, stop-loss rate θ is set to 0.05 and φ, ϕ set to 0, 0.05 respectively. In Table 5 we can see when σ is 0.015 the profit will be highest.

5 Conclusion

We have proposed a method to make stock transact decisions and implement a stock decision support system. The experimental results show that our system is able to provide relatively accurate decision for stock trading, which can bring investors reliable returns. Its success benefit from the following two reasons. The first one is attributed to the in-depth learning ability of DBNs. The second one is oscillation box theory's contribution, which provide a perfect trading strategy and reduces the impact of uncertainly and noise in the stock market. Similarly, the use of gray relation degree method, to a certain extent, helps the DBNs obtain more precise results.

The system introduces DBNs method to predict stock prices time-series can achieve high accurate performance. However, DBNs cost a lot of time to learn the historical data. For this reason, the speed is a bottleneck restricting the system. Therefore, how to improve the system speed and enable it to online decision becomes the future work.

References

[1] S. Chen and P. M. Grant, A clustering technique for digital communications channel equalization using radial basis function networks, IEEE Transactions on Neural Networks, vol. 4, pp. 570-578, 1993.

[2] S. A. George and P. V. Kimon, Forecasting stock market short-term trends using a neuro-fuzzy based methodology, Expert Systems with Applications, vol. 36, no. 7, pp. 10696-10707, 2009.

[3] T. Z. Tan, C. Quek and N. G. See, Biological brain-inspired genetic complementary learning for stock market and bank failure prediction, Computational Intelligence, vol. 23, no. 2, pp. 236-261, 2007.

[4] C. Q. Li, X. P. Liu, S. P. Xu, Stock price prediction based on general regression neural network and fly optimization algorithm, Journal of Computational Information Systems, vol. 8, no. 17, pp. 7021-7028, September 1, 2012.

[5] X. Zhang, H. Fuehers and P. A. Gloor, Predicting asset value through twitter buzz. Advances in Collective Intelligence 2011, pp. 23-34, 2012.

[6] H. J. Liu, H. M. Chen and Y. R. Hu, Financial characteristics and prediction on targets of M & A based on SOM-Hopfield neural network. Industrial Engineering and Engineering Management, 2007 IEEE International Conference on, pp. 80-84, 2007.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, Proc. 26th Int. Conf. Machine Learning, pp. 609-616, 2009.

[8] Bengio, Yoshua, Learning deep architectures for AI, Foundations and trends in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.

[9] G. E. Hinton, S. Osindero, and Y. Teh, A fast learning algorithm for deep belief nets, Neural computation, vol. 18, pp. 1527-1554, 2006.

[10] I. Arel, D. C. Rose, and T. P. Karnowski, Deep Machine Learning–A New Frontier in Artificial Intelligence Research, IEEE Computational Intelligence Magazine, pp. 13-18, 2010.

[11] D. Nicolas, How I made two million dollars in the stock market, BN Publishing.

[12] J. L. Deng. Control problems of gray systems, System Control Letter, vol. 1, no. 4, pp. 288-294, 1982.

[13] L. J. Zhang and Z. J. Li, Gene selection for classifying microarray data using grey relation analysis. Discovery Science, pp. 378-382, 2006.

[14] P. Martin, Technical analysis explained (4th ed.), Paperback. McGraw-Hill Company. ISBN007122 6699.