Journal of Mathematical Research & Exposition
Vol.23, No.1, 83–97, February, 2003

Prediction of Stock Market by BP Neural Networks with Technical Indexes as Input *

LI Zheng-xue[1], *WU Wei*[1], *GAO Wei-dong*[2]

(1. Dept. of Math., Dalian University of Technology, Liaoning 116024, China;

2. Dept. of Comp. Sci & Tech., University of Petroleum, Beijing 102200, China)

Abstract: Some widely-used technical indexes of stock analysis are introduced as input of BP neural networks for the prediction of ups and downs of stock market, and better accuracy of prediction is achieved. A jump training strategy and three varying training ratio methods are used to accelerate the training iteration. An online prediction strategy is applied to monitor the training iteration procedure. The ratio of central distances of prediction examples is defined, in order to locate the un-stable prediction examples.

Key words: BP neural network; stock market; prediction.

Classification: AMS(2000) 62P20,68W/CLC number: TB114, TP391

Document code: A **Article ID**: 1000-341X(2003)01-0083-15

0. Introduction

Stock market is complex and liable to change. The problem that investors concern most is the prediction of ups and downs of stock market. Such traditional techniques as K-line diagram and average line diagram have gotten favored and been common tools for studying stock market due to their simplicity and intuitiveness. However, the study of diagram tendency and the statistical analysis have to be done by human brains, which is a very difficult job even for financial experts. An artificial neural network which is an important tool to simulate the structure and function of the neural cells of human brains may help people to grasp the variation of stock market objectively and accurately.

One of the characteristics of a neural network is its learning (or training) ability. By training, the neural network can give correct answers not only for learned examples, but also for the models similar to the learned examples, showing its strong associative ability and rational ability which are suitable for solving large, nonlinear, and complex classification and function approximation problems. There are many training algorithms for neural networks, of which BP network is well-known for its solid theory and wide application. Many researchers have applied it to stock market to improve the existing

*Received date: 2001-02-27
Foundation item: Supported by the National Natural Science Foundation of China (19971012)
Biography: LI Zheng-xue (1962-), male, Ph.D., born in Heilongjiang province, Associate Professor.

analysis methods (see [5], [6]). For example, Wu, Chen and Liu (cf. [1]) adopted a ordinary BP algorithm to study the up-down situation of the exchange index of Shanghai stock market. Since only some basic datum indexes in stock market were used to construct example vectors, the prediction results there were not ideal. The purpose of the paper is to consider the similar problem and to improve this result by making use of some more effective long-term technical indexes.

This paper is organized as follows. The BP network is introduced in Section 1, where the number of nodes of each layer and a few parameters in the network are settled. The procedures and methods of network training and predicting are given in Section 2. In order to improve the convergence speed of the network training, the network is trained by two new algorithms in Section 3. Numerical experiment results are analyzed in Section 4. Finally, conclusions are given in Section 5.

1. BP Network

1.1 An introduction of BP network

A BP network with a hidden layer can approximate with arbitrary precision an arbitrary nonlinear function defined on a compact set of R^n (cf. [7], [8] and [9]). The topological architecture of a BP network is shown in Fig.1. BP algorithm is a training algorithm with teachers, whose training procedures are divided into two parts: a forward propagation of information and a backward propagation (BP) of error. The network's training procedures is described below.

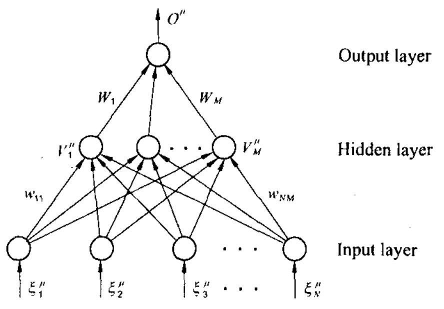

Fig.1 A BP neural network whose output node number is 1

Let the node numbers of input and hidden layers be N and M respectively. In this paper, the node number of the output layer is ascertained as 1. Let the input example vectors be $\xi^\mu = (\xi_1^\mu, \xi_2^\mu, \cdots, \xi_N^\mu)$ $(1 \le \mu \le P)$. We define $\xi_1^\mu = -1$, so as to convert the threshold value for each hidden node into the first component of the weight and to make the mathematical treatment more convenient (see e.g. [9]). Denote by $w_{ij}(1 \le i \le N, 1 \le j \le M)$ the weight connecting the ith input node and the jth hidden node. Denote by $W_j(1 \le j \le M)$ the connection weight between the jth hidden node and the output node. $g(x)$ and $f(x)$ are the activation functions of the hidden layer and the output layer respectively. When training examples ξ^μ are input to the network, the input and output values of the jth hidden node are denoted as h_j^μ and V_j^μ $(1 \le j \le M, 1 \le \mu \le P)$ respectively, while the input values and output values of the output unit are denoted by H^μ and $O^\mu(1 \le \mu \le P)$ respectively. In symbol we have

$$h_j^\mu = \sum_{i=1}^{N} w_{ij}\xi_i^\mu, \tag{1.1}$$

$$V_j^\mu = g(h_j^\mu), \tag{1.2}$$

$$H^\mu = \sum_{j=1}^{M} W_j V_j^\mu, \tag{1.3}$$

$$O^\mu = f(H^\mu). \tag{1.4}$$

Let the desired output corresponding to the input example ξ^μ be ζ^μ. (In classification problem, according to the type of output layer's activation functions, usually ζ^μ are chosen as $\{0,1\}$ or $\{-1,1\}$, and the last definition is chosen in this paper). Then the square error function for this step of training is

$$E_\mu = \frac{1}{2}\left(\zeta^\mu - O^\mu\right)^2. \tag{1.5}$$

The overall square error function after all examples are used is

$$E = \frac{1}{2}\sum_{\mu=1}^{P}\left(\zeta^\mu - O^\mu\right)^2. \tag{1.6}$$

Let W denote the vector containing all the weights. The purpose of BP algorithm is to choose W so as to minimize the error function by, say, the gradient descent method. So the general expression of the iteration formula is

$$W(t+1) = W(t) + \triangle W(t), \tag{1.7a}$$

where

$$\triangle W(t) = \eta\left(-\frac{\partial E}{\partial W}\right)\Bigg|_{W=W(t)} \tag{1.7b}$$

is the weight increment at time t, and the positive constant η is the training ratio.

In practical application, a momentum term is often added to Formula (1.7) to accelerate the convergence speed, resulting in

$$\triangle W(t) = \eta\left(-\frac{\partial E}{\partial W}\right)\Bigg|_{W=W(t)} + \alpha\triangle W(t-1), \tag{1.8}$$

where the positive constant α is a momentum factor.

A popular variation of the standard gradient method (1.7) is a so called *online gradient method* (OGM for short), where we replace E in (1.7) by E_μ. This means that the weight values are modified as soon as a training example is input to the network. Now we have

$$\triangle W_j = \eta\left(-\frac{\partial E_\mu}{\partial W_j}\right) = \eta\left(\zeta^\mu - O^\mu\right) f'\left(H^\mu\right) V_j^\mu. \tag{1.9}$$

By the chain rule and (1.1)-(1.5), we have

$$\triangle w_{ij} = \eta\left(-\frac{\partial E_\mu}{\partial w_{ij}}\right) = \eta\left(\zeta^\mu - O^\mu\right) f'\left(H^\mu\right) W_j g'\left(h_j^\mu\right)\xi_i^\mu. \tag{1.10}$$

The training examples $\{\xi^\mu\}$ are usually supplied to the network in a stochastic order (for example, 2-3-1-1-3-2-$\cdots$).

It is more likely for OGM to jump off from a local minimum of the error function, compared with the standard gradient method, and it requires less memory space. Therefore, OGM is widely used in neural network training (cf. [11]).

After weight values W_j and w_{ij} are determined through network training, we supply to the network with an input vector ξ with respect to a certain day's market situation, resulting in an output value O according to (1.1)-(1.4), which predicts the up ($O \in (0,1)$) or down ($O \in (-1,0)$) of the market in the next day.

As in [1], this paper also uses a three-layer BP network with OGM to study the stock market. The activation function of the output layer is a hyperbolic tangent function $f(x) = \tanh x$, while the activation function of the hidden layer is $g(x) = \tanh \beta x$ ($0 < \beta \leq 1$). In order to have more adjustable parameters and get more precise solutions, a threshold value is set in the hidden layer whose input component is -1. Usually small initial weight values (including threshold value) are chosen, otherwise the activation function will reach saturation in the beginning of training and the network will fall into local minimum around the initial point. Based on numerical experiments, initial weight values are chosen randomly in the interval $(-0.05, 0.05)$.

Remark Unless otherwise stated, all the numerical experiment results in this paper are average values of five independent experiment results under certain conditions.

1.2 training example vectors

1.2.1 The selection of input example vectors

For the convenience of comparing with the results in [1], we also make use of 135 data of the exchange index of Shanghai Stock Market from June 26, 1998 to Jan. 12, 1999. We use the data of the first three months to form 80 training examples, and the data of the last month to form 25 test examples.

A very important problem is to select some suitable technical indexes as the components of the examples. Representativeness is demanded when the indexes are selected, and at the same time, the independence of the indexes should be noticed. Not only should the important effects of recent data be taken into account, but also the effects of historical data should be utilized. Twelve common technical indexes of stock market as given in Table 1 are selected through many experiments.

Table 1 The components of input example vectors

ξ_1	-1	ξ_8	Index K
ξ_2	Today's closing quotation index	ξ_9	Index D
ξ_3	Today's up-down value	ξ_{10}	RSI
ξ_4	Yesterday's up-down value	ξ_{11}	DIF
ξ_5	The day before yesterday's up-down value	ξ_{12}	DEA
ξ_6	Average up-down value in the past ten days(including today)	ξ_{13}	BIAS
ξ_7	Average up-down value in the past thirty days (including today)		

In the above table, components $\xi_1 - \xi_7$ are easy to be understood. The other six indexes are introduced below.

Stochastic (KD line)

Stochastic index line (KD line) is a common technical analysis tool in European and American futures and stock markets. It synthesizes the concept of momentum and the advantages of strong-weak index and moving average line, so it is sensitive to middle and short term market situation. First we need to calculate

$$\text{RSV} = \frac{\text{today's closing quotation price-the lowest price in the past nine days}}{\text{the highest price in the past nine days-the lowest price in the past nine days}} \times 100.$$

Then the values K and D may be obtained:

$$\text{Current day's K} = \text{the past day's K} \times 2/3 + \text{current day's RSV} \times 1/3,$$
$$\text{Current day's D} = \text{the past day's D} \times 2/3 + \text{current day's K} \times 1/3.$$

The values K and D are always between 0 and 100. The past day's values are taken as 50 to start the calculation.

Relative strength index (RSI)

Relative strength index (RSI) is applied first to futures market where fast price variation and intense speculation are common. Later it is introduced into stock market. RSI analyzes the stronger or the weaker trend of buyers and sellers in stock market in a certain period of time according to the extent of ups or downs of stock prices. The period for RSI may be longer or shorter. The shorter the period is, the more sensitive RSI is. In this paper RSI's period is chosen to be twelve days, so the formula is

$$\begin{aligned}\text{RSI} =& \text{average up of closing quotation in the past 12 days} \div \\ & (\text{average up of closing quotation in the past 12 days} + \\ & \text{average down of closing quotation in the past 12 days}) \times 100.\end{aligned}$$

RSI's value is also between 0 and 100. RSI is usually applied together with KD lines for the middle or short term prediction of the market.

DIF and DEA

We first need to calculate EMA (exponential moving average value) which is a development of moving average line. It keeps the advantages of moving average line, but overcomes the defects of moving average line that false signals occur frequently. Based on our numerical experiments, we adopt the twelve and the twenty-six days EMA:

$$\text{EMA12} = \text{the last day's EMA} \times 11/13 + \text{the current day's closing quotation price} \times 2/13,$$
$$\text{EMA26} = \text{the last day's EMA} \times 25/27 + \text{the current day's closing quotation price} \times 2/27.$$

The first day's EMA to start the calculation is taken as the current day's closing quotation price.

Now we can compute DIF and DEA.

$$\text{DIF} = \text{EMA12} - \text{EMA26},$$
$$\text{DEA} = \text{the past day's DEA} \times 25/27 + \text{current day's DIF} \times 2/27.$$

DIF means the difference between the fast moving average line and the slow moving average line, while DEA is the average value of DIF in some days. DIF and DEA run around zero-axis, reflecting not only the wave of the present stock prices, but also the variation trend of the market in future. Usually they are used as middle-term indexes. They may also be used to estimate short-term's rollback point.

BIAS

BIAS is often simply called Y, which means the percent ratio of the stock index deviation from the moving average line. Like moving average line, there are long, middle or short BIASs according to different time periods. A twelve days period is adopted in this paper, so the formula is

$$Y = \left(\frac{\text{current day's closing quotation price}}{\text{average closing quotation price in the past 12 days}} - 1\right) \times 100\%.$$

BIAS might be positive, negative or zero when the stock price is at the moment moving above, below or on the average line, respectively. BIAS increases or decreases with the up or down of stock price tendency.

1.2.2 Preprocessing of input example vectors

The magnitude of the components of the example may differ greatly in practice. If the original data are input directly, the network might be controlled very quickly in training process by the "big" components. So the data have to be preprocessed to make a balance. Based on numerical experiments, the value range of each component in our input example vectors is chosen as $(-3, 3)$, and all the components are normalized accordingly.

1.3 Determination of neuron numbers and parameters

1.3.1 Number of hidden nodes

The function of hidden nodes is to extract the features of input vectors in training process. The choice of its number is very important. But there is no general formula for it. A common method is to try to search through numerical experiments of the practical problems. To do this, we randomly choose 65 training examples from 80 training examples with the other 15 examples as test examples. The parameters of the network are chosen as follows: training ratio $\eta = 0.025$, momentum factor $\alpha = 0.2$, parameter $\beta = 0.65$, overall number of iterations is 1500. Related data are stored when the largest correct prediction ratio of the test is attained. The investigating range of hidden node number is from 1 to 5.

The correct prediction ratio of the test always oscillates between 70 and 80 when number of hidden nodes is 1 or 2, so that 1 and 2 are not taken into account.

Table 2 Network performance for different hidden node numbers

Number of hidden nodes	Number of iterations	Correct prediction in training(%)	Training error	Correct prediction in test(%)	Test error
3	932	94.15	7.298	64	10.306
4	1198	96	5.199	53.333	13.954
5	1206	98.15	2.526	54.667	12.749

Note: the training errors and the test errors in the table (and in the other tables below) are calculated according to (1.6).

When the number of hidden nodes is 4 or 5, the network behaves good for the training examples, but bad for the test examples, indicating an over-training. A much better balance is achieved when the number of hidden nodes is 3. So 3 is chosen as the number of hidden nodes. Thus the numbers of each layer's neurons in the network are 13-3-1.

1.3.2 Determination of parameters

The network has three constant parameters to be chosen: the training ratio η, the momentum factor α and the shape parameter β of the activation function of the hidden nodes.

The training ratio η is one of the key parameters that affect network's behaviour. We can fix the training ratio to be a very small constant, or make it adaptive in the training process in a certain fashion. The former has a stable training process but a long training time. The latter has a fast convergence speed, but is liable to oscillate. The introduction of the momentum term may accelerate the passing through of the error surface's flat region. The shape factor β makes the activation function more "flat" around the origin and enlarges the unsaturated interval of the activation function, so as to improve the efficiency of network training.

The three parameters are selected as follows.

First, we set $\alpha = 0$ and $\beta = 1.0$, and perform 1500 iterations. The errors with respect to the number of iterations are recorded for different η(see Fig. 2). We find that the convergence speed is faster when $\eta = 0.025$. Also note that this η is small, and unlikely to cause oscillation. So we take $\eta = 0.025$.

Next, we set $\eta = 0.025$, $\beta = 1.0$, and check for different α. Several main performance indexes of the network are recorded when the corrective ratio of the training achieves the largest.

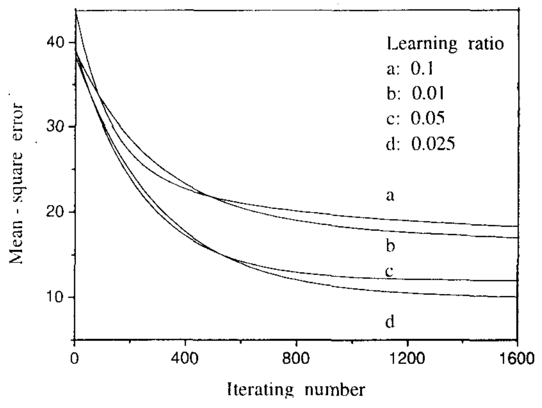

Fig.2 The influence of training ratio

Table 3 **Performance for different α**

α	Number of Iterations	Corrective ratio of training(%)	Training error	training time(s)
0.1	732	93	12.766	4.37
0.15	734	92	13.251	4.22
0.2	704	92.25	13.164	4.07
0.25	794	92.5	13.003	4.60
0.3	705	92.5	12.612	4.06

Obviously, network gives good results when $\alpha = 0.2$ and 0.3. But when $\alpha = 0.3$, there is one training whose corrective ratio is less than 90% in five successive experiments, so that what the table shows are the average values of the other four calculations. If the number of experiments increases, the failing ratio of network training becomes higher when $\alpha = 0.3$. Therefore, we take $\alpha = 0.2$.

Like in the determination of the number of hidden nodes, we randomly choose 65 training examples from 80 examples and leave the other 15 examples as test examples. Suitable β is determined through experiments accordingly.

Table 4 **Experiments for different β**

β	Number of Iterations	Corrective ratio of training(%)	Training error	Corrective ratio of test(%)	Test error
0.35	994	92.5	13.078	68	16.281
0.5	1962	92	14.011	62.4	17.242
0.65	842	92.25	13.397	68.8	15.014
0.8	587	92.75	12.886	68	15.923
0.95	898	91.88	13.408	69	15.449

We see that the test errors are small for β =0.65 and 0.95. When β is 0.95, the corrective ratio of test is a little better, but the other indexes are worse. Besides, since 0.95 is close to 1, it dose not enlarge the unsaturated interval very much. So finally we select $\beta = 0.65$.

2. The network's training and predicting

The purpose of network training is to find weight values with the smallest square error within given number of iterations. After weight values are determined, predicting results may be gotten by inputting predicting example vectors. This kind of prediction may be called as *off-line prediction* since it is done after the network's parameter adjustment has ended and all parameters have been "solidified". Sometimes, training examples and predicting examples have been determined before the network training begins (as in this paper), so *online prediction* may be adopted. Here we interrupt the training process after one network training cycle is finished, and to input predicting examples to the network, then to output onto screen with a scroll manner the predicting results such as number of iterations, corrective ratios of training, training errors, corrective ratios of predicting and predicting errors etc. Thus we can directly observe the training process. In off-line prediction, training and predicting are done separately. Since no predicting example participates in training, only some data relating to training can be gotten, and the predicting results can only be roughly estimated. On the other hand, online prediction, which combines training and predicting, can observe the whole process of training and predicting, and may artificially monitor the tests. When the network falls into local minimum and predicting precision can not attain 50% (which is an unsuccessful experiment), or when the network has attained a desired predicting precision, intervention may be done and training may be terminated. This method may get not only weight values and predicting values but also some useful intermediate results. Over-training (which means higher corrective ratio of training and lower corrective ratio of predicting) may also be prevented. Off-line prediction method is suitable for practical applications, while online prediction method lays particular stress on theoretical study. Table 5 shows the results of 5 successive numerical experiments with online prediction method, while table 6 shows a group of weight values when the corrective ratio of predicting attains 80%. The largest number of iterations is 15000 in the experiments. Corresponding data are recorded when the corrective ratio of training attains the largest. It is obvious that the results are better than the predicting results in reference [1]. It shows that technical indexes are indeed effective for predicting stock market.

Table 5 The predicting results of "online prediction"

Experiment order	Number of Iterations	Corrective ratio of training(%)	Training error	Corrective ratio of test(%)	Predicting error
1	10950	92.5	11.316	80	11.044
2	2123	91.25	12.359	72	13.912
3	1083	92.5	11.986	72	14.208
4	3961	93.75	10.003	68	16.502
5	957	92.5	11.992	68	15.991
Average value	3815	92.5	11.531	72	14.331

Table 6 Optimal weight values(corrective ratio of predicting being 80%)

Hidden node order	1		2		3	
Weight values between input layer nodes and hidden nodes	-3.95741	5.42461	0.78667	-3.54847	0.19966	-2.57813
	1.91568	0.29113	2.44472	2.52897	-2.37693	-1.08246
	-5.14366	4.63912	2.65822	-3.91084	-1.36628	-1.58642
	7.51483	-3.11738	-0.07853	-0.7151	5.45245	1.73193
	-6.2316	-2.22303	3.74178	-1.04297	0.73588	0.50547
	-11.7405	-7.21882	2.08104	-8.23311	-1.49943	-1.26666
	-5.95016		-1.57401		-1.47589	
Weight values between hidden nodes and output layer nodes	6.11536		4.86099		9.94453	

3. Jump training strategy and three variant training ratio algorithms

It may be found from Table 5 that the convergence speed of BP algorithm is slow. In online prediction we observe that when the corrective ratio of training is 92.5% and the corrective ratio of predicting attains 72%, the network training goes into a large flat region.

Only after a long time movement at "a snail's speed" can it attain 80%. If we take the whole number of iterations for the network training as the number of iterations when the corrective ratio of predicting attains the highest 80%, the number of iterations for running across the flat region takes 92.5% of the whole number of iterations. So it is necessary to search fast algorithms in order to improve network's training speed and shorten the time for running across the platform. Many good methods have been proposed, of which two methods are tested in this paper.

3.1 Jump training strategy

Each training example has different convergence speed in training. Some example's square error may become small very early in the process of training. Computing time will be wasted if such examples are repeatedly used for training. The idea of jump training is to define a lower bound $E_{\min}$ for the error. If an example's output error is lower than $E_{\min}$, we do not perform the backward propagation training and jump directly to the next example. Three bounds are set in this paper for the application of jump training. We take $E_{\min} = 0.0001$ when the corrective ratio of training is lower than 72%. The reason for choosing so small an $E_{\min}$ is that the number of jumps should be restricted at the beginning of training to prevent the useful information loses too much. When the corrective ratio of predicting reaches 72% and tends to stabilize (for example, the corrective ratios of predicting do not change within 20 successive training cycles), $E_{\min}$ is enlarged to 0.001. When the corrective ratio of predicting reaches 76%, we take $E_{\min} = 0.01$ in order that the network can run quickly across the platform region. Table 7 shows the training situations of jump training for some experiments when the corrective ratios of predicting are larger than or equal to 76%. For the convenience of comparisons, we give corresponding calculating results of ordinary BP algorithms.

Table 7 Performance comparisons of jump BP and ordinary BP

Algorithm	Number of Iterations	Cost time(s)	Corrective ratio of training(%)	Training error	Corrective ratio of predicting(%)	Predicting error
Ordinary BP	11144	104.4	92.50	10.958	76.8	12.352
Jump BP	6588	55.2	90.25	14.398	76.8	11.729

It may be seen that the iteration number of the weight values decrease about 42% (which occurs mainly in the middle and latter periods of network training) when jump training is adopted. Although the jump training makes the corrective ratios of network training decrease and the training errors enlarged both slightly, the numbers of iterations are decreased and training time is shortened both greatly, showing the feasibility of this method.

3.2 Batch training with varying training ratios

In addition to the jump training, we also train networks by three algorithms with varying training ratios based on batch training which were considered respectively in [2], [3] and [4].

The training ratio formula in a fast training algorithm given by Vogl [2] was

$$\begin{cases} \eta(t) = \lambda\eta(t-1),\ \alpha = \alpha; & \text{for } \triangle E(t) < 0; \\ \eta(t) = \varepsilon\eta(t-1),\ \alpha = 0; & \text{for } \triangle E(t) > 0, \end{cases} \tag{3.1}$$

where $\triangle E(t) = E(t) - E(t-1)$, $E(t) = \sum\limits_{k=0}^{N-1} E_k(t)$, $\lambda > 1$, $0 < \varepsilon < 1$. This method was based on an idea in optimization theory: training ratios were enlarged when global errors decreased; otherwise training ratios were decreased. In our numerical experiments, we take initial training ratio $\eta(0) = 0.6$, momentum factor $\alpha = 0.95$, parameters $\lambda = 1.15$, and $\varepsilon = 0.8$.

The training ratio given by Lei Ming et al.[3] is

$$\eta(t) = e^{\lambda\cos\theta}\eta(t-1), \tag{3.2}$$

where the proportion constant $\lambda \in (0.1, 0.2)$, θ is the angle between the current training error gradient vectors and the last time's training error gradient vectors, and $\cos\theta$ reflects the curvature variation on the error hypersurface. When $\cos\theta > 0$, the error surface is in a flatter region, indicating that the training ratio should be enlarged. When $\cos\theta < 0$, the error surface is in a valley, and the training ratio should be decreased to prevent oscillation. In our case, we take training ratio $\eta(0) = 0.6$, proportion constant $\lambda = 0.15$, and momentum factor $\alpha = 0.85$.

A fast BP training algorithm with adaptive training ratio via linear reinforcement given by Deng Zhidong et al.[4] makes use of gradient information:

$$\begin{cases} \triangle\eta(t) = \varepsilon\lambda\eta(t-1); \\ \eta(t) = \eta(t-1) + \triangle\eta(t), \end{cases} \tag{3.3}$$

where the constant $\varepsilon \in (0.2, 0.3)$, and $\lambda = \operatorname{sgn}\left(\frac{\partial E}{\partial W(t)} \cdot \frac{\partial E}{\partial W(t-1)}\right)$. Its essential idea is that if the angle of the gradient directions of two successive iterations is larger than $\pi/2$, then the training ratio is too large, and should be decreased; otherwise the training ratio should be enlarged. We modify (3.3) a little bit in our application:

$$\begin{cases} \eta(t) = 1.15\eta(t-1), & \text{for } \lambda > 0; \\ \eta(t) = 0.8\eta(t-1), & \text{for } \lambda \le 0, \end{cases} \tag{3.4}$$

where $\eta(0) = 0.6$, momentum factor $\alpha = 0.85$.

All the above three methods are algorithms with adaptive varying training ratio. In order to prevent overflow of the ratio, we require an upper bound of training ratios: $\eta_{\max} = 2.5$.

Some numerical experiments have been done for the above three methods. Table 8 records the experiment results when the corrective training ratio of predicting is large than or equals to 76%.

Table 8 Performance of the three methods with varying training ratio

Algorithm	Number of iterations	Cost time(s)	Corrective ratio of training(%)	Training error	Corrective ratio of predicting(%)	Predicting error
Reference [2]	1192	9.56	90.75	17.435	76.8	12.491
Reference [3]	1450	11.71	90.75	15.624	76.8	12.161
Reference [4]	2621	12.27	93	13.538	76	12.717

Compared with Table 7, the network's training time for the three methods with varying training ratios is indeed shortened. But at the same time we find for all the three training methods that fewer experiments reaches 76% (or 80%) corrective ratio of training in several (say, ten) successive numerical experiments, and that the average values of largest corrective ratios of predicting in each experiment are lower, compared with ordinary BP.

4. Analysis of numerical experiment

4.1 On the randomness

In the process of network training, initial weights are given randomly, and input example vectors are also supplied to the network in a stochastic order. Thus each numerical experiment starts from different initial point and goes through different route, so the searching range on the error surface is enlarged and the chance to get an optimal solution is increased.

To show this in our case, we design a few experiments as follows for comparison: all numerical experiments use the same group of initial weights and the training examples are input to the network.in a fixed order. Then the network's training and predicting are generally worse than that of the above stochastic approach. Therefore, it seems that the introduction of stochastic mechanism is necessary when OGM is applied to network training.

4.2 Classifying predicting examples

Through monitoring the classifying process of the online prediction, we find that the each predicting example behaves differently in the network training. Most predicting examples' predicting results are basically determined after training has begun only for a shorter time. The other fewer predicting examples' predicting results often change in the process of network training. In addition to this, it is found from the 10 numerical experiments with 80% of corrective ratio of predicting that when the corrective ratio of predicting attains 80%, 20 examples are classified correctly in every time, but the other 5 predicting examples are classified incorrectly. It is desirable if there is a method to do the following: 1. Predicting examples are roughly classified according to their different behaviour; 2. The rough positions of these incorrectly classified examples are determined in the whole group of predicting examples when the corrective ratio of predicting attains, say, 80%. To this end, we propose a concept of center distance ratio of predicting examples.

Denote the training example vectors by $T_\mu = (\xi_1^\mu, \xi_2^\mu, \cdots, \xi_{13}^\mu)$ $(1 \le \mu \le 80)$, and the predicting example vectors by $P_\nu = \left(\xi_1^{\nu+80}, \xi_2^{\nu+80}, \cdots, \xi_{13}^{\nu+80}\right)$ $(1 \le \nu \le 25)$. If we set $\overline{T}[k] = \sum_{\mu=1}^{80} \xi_k^\mu / 80$, $\overline{P}[k] = \sum_{\nu=1}^{25} \xi_k^{\nu+80}/25$, and $1 \le k \le 13$, then the vectors $T_0 =$

$\left(\overline{T}[1],\overline{T}[2],\cdots,\overline{T}[13]\right)$ and $P_0 = \left(\overline{P}[1],\overline{P}[2],\cdots,\overline{P}[13]\right)$ which consist of the average value of each example vector's component may be called as the centers of the training example vector group and the predicting example vector group respectively. We define the norm of $x \in R^{13}$ by $\|x\| = \sum_{k=1}^{13} |x_k|$. Then we write

$$\lambda_\nu = \sum_{k=1}^{13} \left| \xi_k^{\nu+80} - \overline{T}[k] \right| \Big/ \sum_{k=1}^{13} \left| \xi_k^{\nu+80} - \overline{P}[k] \right|, \quad 1 \le \nu \le 25. \tag{4.1}$$

Thus the parameter λ_ν means the ratio of the distances between predicting example vector P_ν and vector T_0 and between P_ν and P_0. It is simply called the center distance ratio of predicting example P_ν.

A numerical experiment we did in this respect is as follows. First, 25 predicting examples are numbered with 1-25 in their natural time order. Then, since network training in the beginning is unstable and its results' credibility is lower, we start the monitoring of the training in the middle and latter periods. We concentrate on those trainings which the corrective ratios of predicting attain 80%. When the corrective ratio of predicting attains 64% and tends to be stable, the 25 examples' predicting results are recorded respectively: 1 denotes a correct prediction, and 0 a wrong prediction. Because corrective ratios of predicting increase progressively with a step of 4%(1/25), when the corrective ratio of predicting increases from 64% to 80%, each predicting example may get 5 records. In order to increase reliability of the experiments, 10 numerical experiments in the same conditions are done, so each predicting example getting 50 records, and the value of each record is either 1 or 0. Obviously, the predicting results of the predicting examples whose accumulated sums are close to 50 (or 0) become basically *stable* (unchanged) after the corrective ratios of predicting attain 64%, so that they can be called as *stable predicting examples.* The accumulated sums of the predicting results of the other predicting examples are farther from 0 or 50 (generally between 20 and 40). In the process of predicting, their predicting results often change with the increase of the corrective ratios of predicting, so that this type of predicting examples may be called as *unstable predicting examples.*

Table 9 gives all the examples' center distance ratios λ_ν and the accumulated sums of 50 numerical experiment records.

Table 9 Performance indexes of predicting examples

Example order	12	13	9	11	14$^\times$	19	15	16	
Center distance ratio	3.912	3.861	3.679	3.650	3.494	3.084	3.074	3.058	
Accumulated sum	49	50	49	50	0	50	50	50	
Example order	8	10	21$^\times$	22$^\times$	18	17$^\times$	20	7	
Center distance ratio	3.009	2.994	2.935	2.927	2.906	2.749	2.711	2.390	
Accumulated sum	46	50	0	1	50	0	50	49	
Example order	23	3	6	2	4$^\times$	1	5	24	25
Center distance ratio	2.376	1.815	1.672	1.501	1.478	1.262	1.195	1.007	0.845
Accumulated sum	20	20	50	50	29	30	36	21	50

It is easy to see that the values λ_ν of stable predicting examples are generally larger, which mainly exist in the first two rows in Table 9; while the values λ_ν of unstable predicting examples are less, which is in the last row in the table. In addition, we have recorded the variations of the predicting results of unstable predicting examples by means of monitoring in the process that the corrective ratio of predicting increases from 64% to 80%.

Table 10 Variations of the predicting results of "unstable predicting examples"

Corrective ratio of predicting (%)	Statistics of the variations of predicting results
64-68	8(4), 5(6)
68-72	1(10)
72-76	4(9), 24(9), 23(8)
76-80	3(8)

Note: 8(4) in Table 10, for instance, means that the predicting result of the 8th predicting example has changed for four times in 10 numerical experiments in the process that the corrective ratio of predicting increases from 64% to 68%.

Obviously, in the middle and latter periods of network training, predicting examples' classification is mainly done in descending order of accumulated sums of unstable predicting examples.

It might have been noticed by careful readers that in Table 9 the right upper corners of five examples are remarked with "$\times$". These five examples are those that are always incorrectly classified in the whole 10 numerical experiments mentioned before when the predicting ratio of predicting attains 80%.

The above analyses on the predicting results are not only a kind of "afterprocessing", but also a continuation of prediction. They may reveal quite valuable information of the training. In our problem, we have found the regions where the incorrectly classified examples are concentrated (the middle part of Row 4 in Table 9) by means of the concept of center distance ratio of predicting examples. Based on this, the predicting results (0 or 1) of all the predicting examples in this region (21, 22, 18, 17) are simply taken reverse operations (0 and 1 exchange each other). Then the corrective ratio of predicting increases from 80% to 88%. the network predicting precision is thus improved. Of course, further investigations are needed in this respect.

5. Conclusions

(1) Suitable technical indexes of the stock market are used to construct example vectors, which has increased the corrective ratio of predicting of BP network for the stock market.

(2) Online prediction method we introduced in this paper can help us to monitor the training and predicting processes, and the training may be terminated according to the practical situations so as to improve the efficiency.

(3) Ordinary BP network has good results on the prediction of the of stock market, but its convergence speed is low. The introduction of jump training increases the training speed. Although the three fast batch training algorithms with varying training ratio tested in this paper can increase the training speed, their whole predicting results are a little worse and their stabilities are worse than ordinary BP.

References:

[1] WU Wei, CHEN Wei-qiang, LIU Bo. *Prediction of ups and downs of stock market by BP neural networks* [J]. Journal of Dalian University of Technology, 2001, **41**(1): 9–15.

[2] JIAO Li-cheng. *Neural Network Calculation* [M]. Xidian University Xi'an China Press, Xi'an, 1993, 41–42.

[3] LEI Ming, WU Ya, YANG Shu-zi. *Non-linear time series modelling and forecasting using the neural network approach* [J]. J. Huazhong Univ. of Sci. Tech., 1993, **21**(1): 47–52.

[4] DENG Zhi-dong, SUN Zeng-qi. *Fast BP learning algorithm with adaptive variable stepsize via linear reinforcement* [J]. Pattern Recognition and Artificial Intelligence, 1993, **6**(4): 319–323.

[5] TAKASHI KIMOTO, KAZUO A. *Stock market prediction system with modular neural networks* [J]. International Joint conference on Neural Network, 1990, **1**(1): 1–6.

[6] KEN-ICHI KAMIJO, TETSUJI T. *Stock price pattern recognition: a recurrent neural network approach* [J]. International Joint conference on Neural Network, 1990, **1**(1): 215–221.

[7] XU Bing-zheng, ZHANG Bai-ling, WEI Gang. *Theory and Application of Neural Networks* [M]. South China University of Technology Press, Guangzhou, 1994.

[8] WAN Ke-jun, WAN Ke-cheng. *The Modeling, Prediction and Cortrol of Neural Networks* [M]. Harbin Engineering University Press, Harbin, 1996.

[9] JIAO Li-cheng. *The Theory of Neural Network System* [M]. Xidian University Xi'an China Press, Xi'an, 1996.

[10] XIE Guo-min and LI Wen-gang. *The Analysing Expert of Stock Market Technology* [M]. Enterprise Management Publishing House, Beijing, 1997.

[11] SIMON H. *Neural Networs: A Comprehensive Foundation* [M]. Tsinghua Universtiy Press and Prentice Hall, Beijing, 2001.

引入技术指标的 BP 网络在股市预测中的应用

李 正 学[1], 吴 微[1], 高 维 东[2]

(1. 大连理工大学应用数学系， 辽宁 大连 116024; 2. 石油大学计算机系， 北京 102200)

摘 要: 本文使用股市分析中常用的一些技术指标构造 BP 网络的输入样本向量，在此基础上，对沪市股指的涨跌进行了预测. 数值实验结果表明，该方法能够提高网络预测的正确率. 使用跳跃学习及三种变学习率、批方式的学习算法对 BP 网络进行了训练，节省了预测时间. 运用“在线预测”的方法对预测过程进行了跟踪. 针对预测样本在预测性能及预测结果方面存在的差异，引入预测样本中心距离比的概念对其进行简单的划分，得到一些富有启发性的结果.