An Adaptive BP Learning Algorithm for Stock Market Prediction

Kin Keung Lai[1 2], Lean Yu [1 3] and Shouyang Wang[3]
[1] Department of Management Sciences, City University of Hong Kong,
83 Tat Chee Avenue, Kowloon, Hong Kong
mskklai@cityu.edu.hk
[2] College of Business Administration, Hunan University, Changsha, 410082, China
[3] Institute of Systems Science, Academy of Mathematics and Systems Science,
Chinese Academy of Sciences, Beijing 100080, China
{yulean, sywang}@amss.ac.cn

ABSTRACT

In this study, a novel adaptive learning algorithm for back-propagation neural network (BPNN) based on optimized instantaneous learning rates is proposed. In this new algorithm, the optimized adaptive learning rates are used to adjust the weight changes dynamically. For illustration and testing purposes the proposed algorithm is applied to stock market prediction.

Keywords: Adaptive learning, BPNN, optimal learning rate, stock market prediction

1. INTRODUCTION

Back-propagation neural network (BPNN) is the most popular class of artificial neural networks (ANNs) which have been widely applied to time series prediction, function approximation, pattern recognition, nonlinear system identification and control problems, etc. The basic learning rule of BPNN is based on the gradient descent optimization method and the chain rule, as initially proposed by Werbos [1] in the 1970s. Since the basic learning rule is based on the gradient descent method, which is known for its slowness and its frequent confinement to local minima [2], many improved BP algorithms are developed such as variable step size, adaptive learning [3-4] and others [5-6]. Generally, these algorithms have an improved convergence property, but most of these methods do not use the optimized instantaneous learning rates. In their studies, the learning rate is set to a fixed value when learning. However, it is critical to determine a proper fixed learning rate for the applications of the BPNN. If the learning rate is large, learning may occur quickly, but it may also become unstable and even will not learn at all. To ensure stable learning, the learning rate must be sufficiently small, but with a small learning rate the BPNN may be lead to a long learning time. Also, just how small the learning rate should be is unclear. In addition, for different structures of BPNN and for different applications, the best fixed learning rates are different.

There are other ways to accelerate the network learning using second-order gradient based nonlinear optimization methods, such as the conjugate gradient algorithm [6] and Levenberg-Marquardt algorithm [7]. The crucial drawbacks of these methods, however, are that in many applications computational demands are so large that their effective use in many practical problems is not viable.

A common problem with the all above mentioned methods is a non-optimal choice of the learning rate even with the adaptive change of the learning rate. A solution is to derive optimal learning rate formulae for BPNN and then allow an adaptive change at each iteration step during the learning process. The resulting algorithm will eliminate the need for a search for the proper fixed learning rate and provide fast convergence.

Due to the highly nonlinearity of neural networks, it is difficult to obtain the optimum learning rate. In this paper, a new method based on matrix and optimization techniques is proposed to derive the optimal learning rate and construct an adaptive learning algorithm. To test the efficiency of the proposed algorithm, an important stock index — Nikkei225 is used. The rest of this work is organized as follows. In Section 2, the proposed adaptive learning algorithm with optimal learning rate is presented. In order to testing the proposed algorithm, Section 3 gives an experiment and reports the results. Finally, the conclusions are made in Section 4.

2. THE ADAPTIVE LEARNING ALGORITHM

Consider a three-layer BPNN, which has p nodes in the input layer, q nodes in the hidden layer and k nodes in the output layer. Mathematically, the basic structure of the BPNN model is described by

$$Y(t+1)=\begin{bmatrix} y_1(t+1) \\ y_2(t+1) \\ \cdots \\ y_k(t+1) \end{bmatrix}=\begin{bmatrix} f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t)+w_{i0}(t))v_{1i}(t)+v_{10}(t)] \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t)+w_{i0}(t))v_{2i}(t)+v_{20}(t)] \\ \cdots \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t)+w_{i0}(t))v_{ki}(t)+v_{k0}(t)] \end{bmatrix}$$

$$=\begin{bmatrix} f_2[\sum_{i=0}^{q} f_1(\sum_{j=0}^{p} w_{ij}(t)x_j(t))v_{1i}(t)] \\ f_2[\sum_{i=0}^{q} f_1(\sum_{j=0}^{p} w_{ij}(t)x_j(t))v_{2i}(t)] \\ \cdots \\ f_2[\sum_{i=0}^{q} f_1(\sum_{j=0}^{p} w_{ij}(t)x_j(t))v_{ki}(t)] \end{bmatrix}=\begin{bmatrix} f_2[V_1^T F_1(W(t)X(t))] \\ f_2[V_2^T F_1(W(t)X(t))] \\ \cdots \\ f_2[V_k^T F_1(W(t)X(t))] \end{bmatrix}$$

$$= F_2[V^T(t) F_1(W(t)X(t))] \quad (1)$$

where $x_j(t)$, $j = 1, 2, \ldots, p$, are the inputs of the BPNN; y is the output of the BPNN; $w_{ij}(t)$, $i = 1, \ldots, q$, $j = 1, \ldots, p$, are the weights from the input layer to the hidden layer; $w_{i0}(t)$, $i = 1, \ldots, q$, are the biases of the hidden nodes; $v_{ij}(t)$, $i = 1, \ldots, q$, $j = 1, \ldots, k$, are the weights from the hidden layer to the output layer; $v_{i0}(t)$ is the bias of the output node; t is a time factor; f_1 is the activation function of the nodes for the hidden layer and f_2 is the activation function of the nodes for the output layer. Generally, the activation function for nonlinear nodes is assumed to be a symmetric hyperbolic tangent function, i.e., $f_1(x) = \tanh(u_0^{-1}x)$, and its derivative is $f_1'(x) = u_0^{-1}[1 - f_1^2(x)]$, $f_1''(x) = -2u_0^{-1} f_1(x)[1 - f_1^2(x)]$, where u_0 is the shape factor of the activation function. Specially, some notations in Equation (1) are defined as follows:

$$X=(x_0, x_1, \cdots, x_p)^T \in R^{(p+1)\times 1}, \quad Y=(y_0, y_1, \cdots, y_k)^T \in R^{k\times 1},$$

$$W=\begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1p} \\ w_{20} & w_{21} & \cdots & w_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ w_{q0} & w_{q1} & \cdots & w_{qp} \end{pmatrix}=(W_0, W_1, \cdots, W_p)\in R^{q\times(p+1)},$$

$$V=\begin{pmatrix} v_{10} & v_{20} & \cdots & v_{k0} \\ v_{11} & v_{21} & \cdots & v_{k1} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1q} & v_{2q} & \cdots & v_{kq} \end{pmatrix}=(V_1, V_2, \cdots, V_k)\in R^{(q+1)\times k},$$

$$F_1(W(t)X(t))=\left(F_1(net_0^H(t)) \quad F_1(net_1^H(t)) \quad \cdots \quad F_1(net_q^H(t))\right)^T \in R^{(q+1)\times 1}$$

$$net_i^H(t)=\sum_{j=0}^{p} w_{ij}(t)x_j(t), \quad i = 0, 1, \ldots, q.$$

Usually, by estimating model parameter vector (W, V) via BPNN training and learning, we can realize the corresponding tasks such as function approximation, system identification or prediction. In fact, the model parameter vector (W, V) can be obtained by iteratively minimizing a cost function $E(X: W, V)$. In general, $E(X: W, V)$ is a sum of the error squares cost function with k output nodes and N training pairs or patterns, that is,

$$E(X:W,V)=\frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{p} e_{ij}^2=\frac{1}{2}\sum_{j=1}^{N} e_j^T e_j$$

$$=\frac{1}{2}\sum_{j=1}^{N}[y_j - y_j(X:W,V)]^T[y_j - y_j(X:W,V)] \quad (2)$$

where y_j is the jth actual value and $y_j(X: W,V)$ is the jth estimated value.

Given the time factor t, Equation (2) can be rewritten as

$$E(t)=\frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{k} e_{ij}^2(t)=\frac{1}{2}\sum_{j=1}^{N} e_j^T(t)e_j(t)$$

$$=\frac{1}{2}\sum_{j=1}^{N}[y_j - y_j(t)]^T[y_j - y_j(t)] \quad (3)$$

where $e_j(t)=[e_{1j}(t) \quad e_{2j}(t) \quad \cdots \quad e_{kj}(t)]^T \in R^{k\times 1}, j=1,2,\cdots,N.$

By applying the steepest descent method to the error cost function $E(t)$ (i.e., Equation (3)), we can obtain the gradient of $E(t)$ with respect to V and W, respectively.

$$\nabla_V E(t)=\frac{\partial E(t)}{\partial V(t)}=\sum_{j=1}^{N}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial e_{ij}(t)}{\partial V(t)}=-\sum_{j=1}^{N}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial y_{ij}(t)}{\partial V(t)}$$

$$=-\sum_{j=1}^{N} e_j(t)F_{2(j)}'[V^T F_{1(j)}(WX)]'=-\sum_{j=1}^{N} F_{1(j)}(WX)e_j^T(t)F_{2(j)}'$$

$$=-\sum_{j=1}^{N} F_{1(j)}e_j^T(t)F_{2(j)}'$$

$$\nabla_W E(t)=\frac{\partial E(t)}{\partial W(t)}=\sum_{j=1}^{N}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial e_{ij}(t)}{\partial W(t)}=-\sum_{j=1}^{N}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial y_{ij}(t)}{\partial W(t)}$$

$$=-\sum_{j=1}^{N} e_j(t)F_{2(j)}'[V^T F_{1(j)}(WX)]'=-\sum_{j=1}^{N} F_{1(j)}'\bar{V}F_{2(j)}'e_j(t)x_j^T(t)$$

$$=-\sum_{j=1}^{N} \bar{F}_{1(j)}'\bar{V}F_{2(j)}'e_j x_j^T$$

So, the updated formulae of weights are given by, respectively

$$\Delta V = -\eta\nabla_V E(t)=\eta\sum_{j=1}^{N} F_{1(j)}e_j^T F_{2(j)}' \quad (4)$$

$$\Delta W = -\eta\nabla_W E(t)=\eta\sum_{j=1}^{N} \bar{F}_{1(j)}'\bar{V}F_{2(j)}'e_j x_j^T \quad (5)$$

where η is the learning rate;

$$\bar{F}_{1(j)}'=\mathrm{diag}[f_{1(1)}' \quad f_{1(2)}' \quad \cdots \quad f_{1(q)}'] \in R^{q\times q},$$

$$f_{1(i)}'=f_1'(net_i^H)=\frac{\partial f_1(net_i^H)}{\partial net_i^H}, i=1,2,\cdots,q,$$

$$\bar{v}_i=[v_{i1} \quad \cdots \quad v_{iq}]^T \in R^{q\times 1}, i=1,2,\cdots,q,$$

$$F_2'=diag[f_{2(1)}' \quad f_{2(2)}' \quad \cdots \quad f_{2(k)}'] \in R^{k\times k},$$

$$\bar{V}=\begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1q} & v_{2q} & \cdots & v_{kq} \end{bmatrix}=[\bar{v}_1 \quad \bar{v}_2 \quad \cdots \quad \bar{v}_k] \in R^{q\times p}$$

$$f_{2(i)}'=f_2'[v_i^T F_1(WX)]=\frac{\partial f_2[v_i^T F_1(WX)]}{\partial[v_i^T F_1(WX)]}, i=1,2,\cdots k$$

To derive the optimal learning rate, let Δ be an increment operator and consider the general error equation:

$$\Delta e(t+1)=e(t+1)-e(t)$$
$$=y - y(t+1) - y + y(t)=-\Delta y(t+1) \quad (6)$$

where $\Delta y(t+1)\equiv 0$. This means there is no change in the pattern during the neural networks' learning

procedure and the change of output of neural networks is

$$\Delta y(t+1) = \eta \xi(t) e(t) \tag{7}$$

where $\xi(t) = \mathbf{F}_2'[(\mathbf{F}_1^T\mathbf{F}_1)\otimes \mathbf{I}_{k^2} + (\mathbf{X}^T\mathbf{X})\times(\overline{V}\mathbf{F}_1'\mathbf{F}_1'\overline{V})]\mathbf{F}_2'$ with

$$\mathbf{F}_2' = \begin{bmatrix} F_{2(1)}' & 0 & \cdots & 0 \\ 0 & F_{2(2)}' & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & F_{2(N)}' \end{bmatrix} \quad \mathbf{X}^T\mathbf{X} = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \cdots & x_1^T x_N \\ x_2^T x_1 & x_2^T x_2 & \cdots & x_2^T x_N \\ \cdots & \cdots & \cdots & \cdots \\ x_N^T x_1 & x_N^T x_2 & \cdots & x_N^T x_N \end{bmatrix}$$

$$\mathbf{F}_1^T\mathbf{F}_1 = \begin{bmatrix} F_{1(1)}^T F_{1(1)} & F_{1(1)}^T F_{1(2)} & \cdots & F_{1(1)}^T F_{1(N)} \\ F_{1(2)}^T F_{1(1)} & F_{1(2)}^T F_{1(2)} & \cdots & F_{1(2)}^T F_{1(N)} \\ \cdots & \cdots & \cdots & \cdots \\ F_{1(N)}^T F_{1(1)} & F_{1(N)}^T F_{1(2)} & \cdots & F_{1(N)}^T F_{1(N)} \end{bmatrix}$$

$$\mathbf{F}_1'\mathbf{F}_1' = \begin{bmatrix} \overline{F}_{1(1)}'\overline{F}_{1(1)}' & \overline{F}_{1(1)}'\overline{F}_{1(2)}' & \cdots & \overline{F}_{1(1)}'\overline{F}_{1(N)}' \\ \overline{F}_{1(2)}'\overline{F}_{1(1)}' & \overline{F}_{1(2)}'\overline{F}_{1(2)}' & \cdots & \overline{F}_{1(2)}'\overline{F}_{1(N)}' \\ \cdots & \cdots & \cdots & \cdots \\ \overline{F}_{1(N)}'\overline{F}_{1(1)}' & \overline{F}_{1(N)}'\overline{F}_{1(2)}' & \cdots & \overline{F}_{1(N)}'\overline{F}_{1(N)}' \end{bmatrix}$$

Here, $\otimes$ indicates a direct product, and $\times$ indicates a cross product.

In order to prove Equation (7), a lemma must be introduced.

Lemma: The total time derivative of the BPNN single output $v^T F_1(WX)$ is given by

$$\frac{d[v^T F_1(WX)]}{dt} = F_1(WX)\frac{dv}{dt} + \overline{v}^T \overline{F_1}'(WX)\frac{dW}{dt}X$$
$$= \frac{dv^T}{dt} F_1(WX) + \overline{v}^T \overline{F_1}'(WX)\frac{dW}{dt}X$$

Proof: Derivation of $\frac{d[v^T F_1(WX)]}{dt}$ is as follows:

$$\frac{d[v^T F_1(WX)]}{dt} = \frac{d\left[\sum_{i=0}^{q} v_i f_1\left(\sum_{j=0}^{p} w_{ij}x_j\right)\right]}{dt}$$
$$= \sum_{i=0}^{q} \frac{\partial\left[\sum_{i=0}^{q} v_i f_1\left(\sum_{j=0}^{p} w_{ij}x_j\right)\right]}{\partial v_i}\frac{dv_i}{dt} + \sum_{i=0}^{q}\sum_{j=0}^{p} \frac{\partial\left[\sum_{i=0}^{q} v_i f_1\left(\sum_{j=0}^{p} w_{ij}x_j\right)\right]}{\partial w_{ij}}\frac{dw_{ij}}{dt}$$
$$= \sum_{i=0}^{q} f_1\left(\sum_{j=0}^{p} w_{ij}x_j\right)\frac{dv_i}{dt} + \sum_{i=0}^{q}\sum_{j=0}^{p} v_i f_1'\left(\sum_{j=0}^{p} w_{ij}x_j\right)x_j\frac{dw_{ij}}{dt}$$
$$= f_1(net_0)\frac{dv_0}{dt} + f_1(net_1)\frac{dv_1}{dt} + \cdots + f_1(net_q)\frac{dv_q}{dt}$$
$$+ v_0 f_1'(net_0)\left[x_0\frac{dw_{00}}{dt} + x_1\frac{dw_{01}}{dt} + \cdots + x_p\frac{dw_{0p}}{dt}\right]$$
$$+ v_1 f_1'(net_1)\left[x_0\frac{dw_{10}}{dt} + x_1\frac{dw_{11}}{dt} + \cdots + x_p\frac{dw_{1p}}{dt}\right]$$
$$+ v_q f_1'(net_q)\left[x_0\frac{dw_{q0}}{dt} + x_1\frac{dw_{q1}}{dt} + \cdots + x_p\frac{dw_{qp}}{dt}\right]$$
$$= \left[f_1(net_0) \quad f_1(net_1) \quad f_1(net_q)\right]\left[\frac{dv_0}{dt} \quad \frac{dv_1}{dt} \quad \cdots \quad \frac{dv_q}{dt}\right]^T$$
$$+ \left[v_1 \quad v_2 \quad \cdots \quad v_q\right]\begin{bmatrix} f'(net_1) & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & f'(net_q) \end{bmatrix}\begin{pmatrix} \text{due to } f(net_0)\equiv 1, \\ f'(net_0)=0 \end{pmatrix}$$
$$\times \begin{bmatrix} \frac{dw_{00}}{dt} & \frac{dw_{01}}{dt} & \cdots & \frac{dw_{0p}}{dt} \\ \frac{dw_{10}}{dt} & \frac{dw_{11}}{dt} & \cdots & \frac{dw_{1p}}{dt} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{dw_{q0}}{dt} & \frac{dw_{q1}}{dt} & \cdots & \frac{dw_{qp}}{dt} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_p \end{bmatrix}$$
$$= F_1(WX)\frac{dv}{dt} + \overline{v}^T \overline{F_1}'(WX)\frac{dW}{dt}X$$
$$= \frac{dv^T}{dt} F_1(WX) + \overline{v}^T \overline{F_1}'(WX)\frac{dW}{dt}X$$

In the following, we start to prove Equation (7). Let us consider the change of output of MLFNN for the *m*th pattern first. The above Lemma together with Equations (4) and (5) gives

$$\Delta y_m(t+1) = \Delta F_2\left[V^T F_1(Wx_m)\right] = \begin{bmatrix} \Delta f_{2(1),m} \\ \Delta f_{2(2),m} \\ \cdots \\ \Delta f_{2(k),m} \end{bmatrix}$$
$$= \begin{bmatrix} f_{2(1),m}'\cdot(F_{1,m}^T\Delta v_1 + v_1^T\overline{F_{1,m}}'\Delta Wx_m) \\ f_{2(2),m}'\cdot(F_{1,m}^T\Delta v_2 + v_2^T\overline{F_{1,m}}'\Delta Wx_m) \\ \cdots \\ f_{2(k),m}'\cdot(F_{1,m}^T\Delta v_k + v_k^T\overline{F_{1,m}}'\Delta Wx_m) \end{bmatrix}$$
$$= \begin{bmatrix} f_{2(1),m}'\cdot(F_{1,m}^T\sum_{j=1}^{N}\Delta v_{1j} + \overline{v}_1^T\overline{F_{1,m}}'\Delta Wx_m) \\ f_{2(2),m}'\cdot(F_{1,m}^T\sum_{j=1}^{N}\Delta v_{2j} + \overline{v}_2^T\overline{F_{1,m}}'\Delta Wx_m) \\ \cdots \\ f_{2(k),m}'\cdot(F_{1,m}^T\sum_{j=1}^{N}\Delta v_{kj} + \overline{v}_k^T\overline{F_{1,m}}'\Delta Wx_m) \end{bmatrix}$$
$$= \begin{bmatrix} f_{2(1),m}'\cdot\left[F_{1,m}^T\left(\eta\sum_{j=1}^{N}F_{1j}e_{1j}f_{2(1),j}'\right) + v_1^T\overline{F}_{1,m}'\left(\eta\sum_{j=1}^{N}\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T\right)x_m\right] \\ f_{2(2),m}'\cdot\left[F_{1,m}^T\left(\eta\sum_{j=1}^{N}F_{1j}e_{2j}f_{2(2),j}'\right) + v_2^T\overline{F}_{1,m}'\left(\eta\sum_{j=1}^{N}\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T\right)x_m\right] \\ \cdots \\ f_{2(k),m}'\cdot\left[F_{1,m}^T\left(\eta\sum_{j=1}^{N}F_{1j}e_{kj}f_{2(k),j}'\right) + v_k^T\overline{F}_{1,m}'\left(\eta\sum_{j=1}^{N}\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T\right)x_m\right] \end{bmatrix}$$
$$= \eta\sum_{j=1}^{N}\begin{bmatrix} f_{2(1),m}'\cdot(F_{1,m}^T F_{1j}e_{1j}f_{2(1),j}' + \overline{v}_1^T\overline{F}_{1,m}'\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T x_m) \\ f_{2(2),m}'\cdot(F_{1,m}^T F_{1j}e_{2j}f_{2(2),j}' + \overline{v}_2^T\overline{F}_{1,m}'\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T x_m) \\ \cdots \\ f_{2(k),m}'\cdot(F_{1,m}^T F_{1j}e_{kj}f_{2(k),j}' + \overline{v}_k^T\overline{F}_{1,m}'\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T x_m) \end{bmatrix}$$
$$= \eta\sum_{j=1}^{N}\begin{bmatrix} f_{2(1),m}' & 0 & \cdots & 0 \\ 0 & f_{2(2),m}' & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & f_{2(k),m}' \end{bmatrix}\begin{bmatrix} F_{1,m}^T F_{1j}e_{1j}f_{2(1),j}' + \overline{v}_1^T\overline{F}_{1,m}'\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T x_m \\ F_{1,m}^T F_{1j}e_{2j}f_{2(2),j}' + \overline{v}_2^T\overline{F}_{1,m}'\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T x_m \\ \cdots \\ F_{1,m}^T F_{1j}e_{kj}f_{2(k),j}' + \overline{v}_k^T\overline{F}_{1,m}'\overline{F}_{1j}'\overline{VF}_{2j}'e_j x_j^T x_m \end{bmatrix}$$
$$= \eta\sum_{j=1}^{N} F_{2,m}'\cdot(F_{2j}'e_j F_{1,m}^T F_{1j} + \overline{V}^T F_{1,m}'F_{1j}'\overline{VF}_{2j}'e_j x_j^T x_m)$$
$$= \eta\sum_{j=1}^{N} F_{2,m}'\cdot(F_{1,m}^T F_{1j}I_{k^2} + \overline{V}^T F_{1,m}'F_{1j}'\overline{V}x_j^T x_m)\cdot F_{2j}'e_j$$

So, the total change caused by all patterns is

$$\Delta y(t+1) = \begin{bmatrix} \Delta y_1(t+1) \\ \Delta y_2(t+1) \\ \cdots \\ \Delta y_m(t+1) \\ \cdots \\ \Delta y_N(t+1) \end{bmatrix} = \begin{bmatrix} \eta\sum_{j=1}^{N}F_{2,1}'\cdot(F_{1,1}^T F_{1j}I_{k^2} + \overline{V}^T F_{1,1}'F_{1j}'\overline{V}x_j^T x_1)\cdot F_{2j}'e_j \\ \eta\sum_{j=1}^{N}F_{2,2}'\cdot(F_{1,2}^T F_{1j}I_{k^2} + \overline{V}^T F_{1,2}'F_{1j}'\overline{V}x_j^T x_2)\cdot F_{2j}'e_j \\ \cdots \\ \eta\sum_{j=1}^{N}F_{2,m}'\cdot(F_{1,m}^T F_{1j}I_{k^2} + \overline{V}^T F_{1,m}'F_{1j}'\overline{V}x_j^T x_m)\cdot F_{2j}'e_j \\ \cdots \\ \eta\sum_{j=1}^{N}F_{2,N}'\cdot(F_{1,N}^T F_{1j}I_{k^2} + \overline{V}^T F_{1,N}'F_{1j}'\overline{V}x_j^T x_N)\cdot F_{2j}'e_j \end{bmatrix}$$

$$= \eta \begin{bmatrix} F'_{2,1} & 0 & \cdots & 0 \\ 0 & F'_{2,2} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & F'_{2,N} \end{bmatrix}$$

$$\times \begin{bmatrix} (F_{1,1}^T F_{1,1} I_{k^2} + \bar{V}^T F_{1,1}^T F_{1,1} \bar{V} x_1^T x_1) & (F_{1,1}^T F_{1,2} I_{k^2} + \bar{V}^T F_{1,1}^T F_{1,2} \bar{V} x_1^T x_2) \\ (F_{1,2}^T F_{1,1} I_{k^2} + \bar{V}^T F_{1,2}^T F_{1,1} \bar{V} x_2^T x_1) & (F_{1,2}^T F_{1,2} I_{k^2} + \bar{V}^T F_{1,2}^T F_{1,2} \bar{V} x_2^T x_2) \\ \cdots & \cdots \\ (F_{1,N}^T F_{1,1} I_{k^2} + \bar{V}^T F_{1,N}^T F_{1,1} \bar{V} x_N^T x_1) & (F_{1,N}^T F_{1,2} I_{k^2} + \bar{V}^T F_{1,N}^T F_{1,2} \bar{V} x_N^T x_2) \end{bmatrix}$$

$$\begin{bmatrix} \cdots & (F_{1,1}^T F_{1,N} I_{k^2} + \bar{V}^T F_{1,1}^T F_{1,N} \bar{V} x_1^T x_N) \\ \cdots & (F_{1,2}^T F_{1,N} I_{k^2} + \bar{V}^T F_{1,2}^T F_{1,N} \bar{V} x_2^T x_N) \\ \cdots & \cdots \\ \cdots & (F_{1,N}^T F_{1,N} I_{k^2} + \bar{V}^T F_{1,N}^T F_{1,N} \bar{V} x_N^T x_N) \end{bmatrix}$$

$$\times \begin{bmatrix} F'_{21} e_1 \\ F'_{22} e_2 \\ \cdots \\ F'_{2N} e_N \end{bmatrix} = \eta \begin{bmatrix} F'_{21} & 0 & \cdots & 0 \\ 0 & F'_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & F'_{2N} \end{bmatrix}$$

$$\left\{ \begin{bmatrix} F_{11}^T F_{11} I_{k^2} & F_{11}^T F_{12} I_{k^2} & \cdots & F_{11}^T F_{1N} I_{k^2} \\ F_{12}^T F_{11} I_{k^2} & F_{12}^T F_{12} I_{k^2} & \cdots & F_{12}^T F_{1N} I_{k^2} \\ \cdots & \cdots & \cdots & \cdots \\ F_{1N}^T F_{11} I_{k^2} & F_{1N}^T F_{12} I_{k^2} & \cdots & F_{1N}^T F_{1N} I_{k^2} \end{bmatrix} \right.$$

$$+ \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \cdots & x_1^T x_N \\ x_2^T x_1 & x_2^T x_2 & \cdots & x_2^T x_N \\ \cdots & \cdots & \cdots & \cdots \\ x_N^T x_1 & x_N^T x_2 & \cdots & x_N^T x_N \end{bmatrix} \times$$

$$\left. \left(\bar{V}^T \begin{bmatrix} \bar{F}'_{11} \bar{F}'_{11} & \bar{F}'_{11} \bar{F}'_{12} & \cdots & \bar{F}'_{11} \bar{F}'_{1N} \\ \bar{F}'_{12} \bar{F}'_{11} & \bar{F}'_{12} \bar{F}'_{12} & \cdots & \bar{F}'_{12} \bar{F}'_{1N} \\ \cdots & \cdots & \cdots & \cdots \\ \bar{F}'_{1N} \bar{F}'_{11} & \bar{F}'_{1N} \bar{F}'_{12} & \cdots & \bar{F}'_{1N} \bar{F}'_{1N} \end{bmatrix} \bar{V} \right) \right\}$$

$$\times \begin{bmatrix} F'_{21} & 0 & \cdots & 0 \\ 0 & F'_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & F'_{2N} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \cdots \\ e_N \end{bmatrix}$$

$$= \eta \mathbf{F}'_2 [(\mathbf{F}_1^T \mathbf{F}_1) \otimes \mathbf{I}_{k^2} +$$

$$(\mathbf{X}^T \mathbf{X}) \times (\bar{V} \mathbf{F}_1' \mathbf{F}_1' \bar{V})] \mathbf{F}'_2 e(t) = \eta \xi(t) e(t)$$

Substituting (7) into (6), we obtain

$$e(t+1) = e(t) - \eta \xi(t) e(t) \tag{8}$$

The objective here is to derive an optimal learning rate η. That is, at iteration t, an optimal value of the learning rate, $\eta^*(t)$, which minimizes $E(t+1)$ is obtained. Define the cost function:

$$E(t+1) = \frac{1}{2} e^T(t+1) e(t+1) \tag{9}$$

Using Equation (8), Equation (9) may be written as

$$E(t+1) = 0.5[e(t) - \eta \xi(t) e(t)]^T [e(t) - \eta \xi(t) e(t)] \tag{10}$$

which gives the error e, at iteration t+1, as a function of the learning rate η, which minimizes $E(t+1)$. Now we use the first and second order conditions

$$\left. \frac{dE(t+1)}{d\eta} \right|_{\eta = \eta^*(t)} = -\frac{1}{2} [\xi(t) e(t)]^T [e(t) - \eta^*(t) \xi(t) e(t)]$$

$$- \frac{1}{2} [e(t) - \eta^*(t) \xi(t) e(t)]^T \xi(t) e(t) = 0$$

$$\left. \frac{d^2 E(t+1)}{d\eta^2} \right|_{\eta = \eta^*(t)} = e^T(t) \xi^T(t) \xi(t) e(t) > 0$$

Since $\xi(t)$ is positively defined, the second condition is met and the optimum value of the learning rate is found to be

$$\eta^*(t) = \frac{e^T(t) \xi^T(t) e(t)}{e^T(t) \xi^T(t) \xi(t) e(t)} \tag{11}$$

Finally, the increments of the BP neural network parameters, by using the optimal learning rate, are obtained by replacing the η^* given by Equation (11) to Equations (4) and (5), which yield

$$\Delta V = -\eta \nabla_V E(t)$$

$$= \frac{e^T(t) \xi^T(t) e(t)}{e^T(t) \xi^T(t) \xi(t) e(t)} \sum_{j=1}^{N} F_{1(j)} e_j^T F'_{2(j)} \tag{12}$$

$$\Delta W = -\eta \nabla_W E(t)$$

$$= \frac{e^T(t) \xi^T(t) e(t)}{e^T(t) \xi^T(t) \xi(t) e(t)} \sum_{j=1}^{N} \bar{F}'_{1(j)} \bar{V} F'_{2(j)} e_j x_j^T \tag{13}$$

Using the new weight update formulae with optimal learning rates, a new learning algorithm is generated. To verify the effectiveness of the proposed adaptive learning model, a major stock index (Nikkei225) is used as testing targets. A detailed process is presented below.

3. EMPIRICAL STUDY

3.1 Data Description

In the experiments, the data of an important stock index (Nikkei225) is daily and is obtained from Datastream. The entire data set covers the period from January 1 2000 to December 31 2004. The data sets are divided into two periods: the first period covers from January 1 2000 to December 31 2003 while the second period is from January 1 2004 to December 31 2004. The first period, which is assigned to in-sample estimation, is used to network learning and training. The second period is reserved for out-of-sample evaluation. For brevity, the original data are not listed in the paper, and detailed data can be obtained from the sources.

To examine the forecasting performance, the root mean squared error (*RMSE*) and directional change statistics (D_{stat}) of stock index movement are employed in this study. The directional change statistics (D_{stat}) can be expressed as

$$D_{stat} = \sum_{t=1}^{N} a_t \Big/ N \tag{14}$$

where a_t =1 if $(x_{t+1} - x_t)(\hat{x}_{t+1} - x_t) \geq 0$, and a_t =0 otherwise.

3.2 Experiment Results

When the data are prepared, we begin to train BPNN model. In these experiments, we prepare 5 years' daily data. We use the first 4 years' daily data to train and validate the network, and use the last one years' data to test the prediction performance. For comparison, the standard three-layer BP neural network is used as benchmark model. This study varies the number of nodes in the hidden layer and stopping criteria for training. In this study, 5, 10, 20 hidden nodes for each stopping criteria because the BP network does not have a general rule for determining the optimal number of hidden nodes. The study uses 500, 1000, 2000 and 4000 learning epochs for the stopping criteria of BPNN. For standard BPNN model, the learning rate is set to 0.25. The hidden nodes use the sigmoid transfer function and the output node uses the linear transfer function. The study allows 5 input nodes in terms of the results of auto-regression testing. The comparison of experiment results are reported in Table 1.

Table 1 The prediction performance comparison of various BPNN models

Stock index	Training epochs	Number of hidden nodes	RMSE		D_{stat}(%)	
			Standard BPNN	Adaptive BPNN	Standard BPNN	Adaptive BPNN
Nikkei 225	500	5	100.3541	70.1124	49.63	59.44
		10	89.6472	54.3589	51.58	63.38
		20	70.5428	41.2547	52.63	62.38
	1000	5	81.5477	50.3584	50.36	58.76
		10	51.8545	39.6874	52.05	61.02
		20	37.5426	21.2387	52.63	64.47
	2000	5	40.3376	14.3541	53.54	66.71
		10	22.5474	8.4579	***55.68***	70.25
		20	***16.3785***	***5.4763***	55.41	***72.39***
	4000	5	35.4754	20.2378	52.24	65.65
		10	36.3687	13.3782	54.35	72.35
		20	21.3523	7.8524	52.63	68.36

As can be seen from Table 1, we can find that (1) the best prediction performance for the testing data is generally produced when the number of hidden nodes is 20 and the training epochs are 2000. (2) Generally speaking, the prediction performance improves with the increase of training epochs and hidden nodes. (3) Usually, too few training epochs and hidden nodes can not lead to a good forecasting result. (4) The performance of the proposed adaptive BPNN model is much better than that of the standard BPNN model in the experiment.

In addition, focusing on two indicators of Table 1, we find the proposed adaptive BPNN model performs much better than the standard BPNN models in all testing cases. These results also indicate the feasibility of the adaptive BPNN model in stock index forecasting.

4. CONCLUSIONS

In this study, an adaptive BP learning algorithms with optimal learning rate is first proposed. And then this exploratory research examines the potential of using an adaptive BPNN model to predict an important international stock index — Nikkei 225. Our empirical results suggest that the adaptive BPNN model may provide better forecasts than the standard BPNN model. The comparative evaluation is based on a variety of statistics such as *RMSE* and D_{stat}. In our empirical investigation, the adaptive BPNN model outperforms the standard BPNN model in terms of *RMSE* and D_{stat}. Furthermore, our experimental analyses reveal that the *RMSE* and D_{stat} for the stock index using the proposed adaptive BPNN model are significantly better than those obtained using the standard BPNN model. This implies that the proposed adaptive BPNN model can be used as a feasible solution for stock market prediction.

ACKNOWLEDGEMENTS

This work is partially supported by National Natural Science Foundation of China; Chinese Academy of Sciences; Key Laboratory of Management, Decision and Information Systems and SRG of City University of Hong Kong (No. 7001806).

REFERENCES

[1] Widrow, B., Lehr, M.A.: 30 Years of Adaptive Neural Networks: Perception, Madaline, and Backprpagation. *Proceedings of the IEEE Neural Networks I: Theory & Modeling* (*Special issue*) 78 (1990) 1415-1442.

[2] Rumelhart, D.E., McClelland, J.L.: *Parallel Distributed Processing*. MIT Press, Cambridge, MA 1986.

[3] Tollenaere, T.: SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties. *Neural Networks* 3 (1990) 561-573.

[4] Park, D.C., El-Sharkawi, M.A., Marks II, R.J.: An Adaptive Training Neural Network. *IEEE Transactions on Neural Networks* 2 (1991) 334-345.

[5] Jacobs, R.A.: Increase Rates of Convergence through Learning Rate Adaptation. *Neural Networks* 1 (1988) 295-307.

[6] Brent, R.P.: Fast Training Algorithms for Multilayer Neural Nets. *IEEE Transactions on Neural Networks* 2 (1991) 346-35.

[7] Hagan, M.T., Menhaj, M.: Training Feedforward Networks with Marquardt Algorithm. *IEEE Transactions on Neural Networks* 5 (1994) 989-993.