INFORMATICA, 2003, Vol. 14, No. 1, 95–110

An Algorithm of Neural Network and Application to Data Processing in Concrete Engineering

Ji-Zong WANG
President Office, Hebei Institute of Architectural Science and Technology
Handan, Hebei, 056038 P.R. China
e-mail: wangjizong@263.net

Xi-Juan WANG
Institute for Reinforced and Prestressed Concrete Structures, Ruhr-University Bochum
Girondelle 78b, Haus 5, 44799 Bochum, Germany

Hong-Guang NI
Beijing Dacheng Real Estimate Development Corporation
No. 28 West Street of Xuan-Wu-Men, Xuan-Wu District, Beijing, 100053 P.R. China

Received: October 2002

Abstract. It is a complex non-linear problem to predict mechanical properties of concrete. As a new approach, the artificial neural networks can extract rules from data, but have difficulties with convergence by the traditional algorithms. The authors defined a new convex function of the grand total error and deduced a global optimization back-propagation algorithm (GOBPA), which can solve the local minimum problem. For weights' adjustment and errors' computation of the neurons in various layers, a set of formulae are obtained by optimizing the grand total error function over a simple output space instead of a complicated weight space. Concrete strength simulated by neural networks accords with the data of the experiments on concrete, which demonstrates that this method is applicable to concrete properties' prediction meeting the required precision. Computation results show that GOBPA performs better than a linear regression analysis.

Key words: data processing, neural networks, global optimization algorithm, properties of concrete.

1. Introduction

Concrete is nowadays one of the most important man-made building materials used in buildings, bridges, tunnel constructions and so on. It is created by proper mixing of coarse and fine aggregates and cement with an adequate and controlled amount of water. One of the most important properties of concrete is its compressive strength. One normally needs a long cure period to measure the 28-day compressive strength of concrete specimens. That makes it difficult to follow the steps of a current construction. To overcome this problem, many experimental methods have been developed to predict concrete strength (Wang and Chen, 1997; Wu and Lian, 1999).

All of the experimental methods can be classified into two categories. The first is to quicken the hardening process which functions by increasing the temperature and pressure of the curing surroundings, and then by measuring the early compressive strength values of concrete at the time of 1-hour, half-day or 3-day. Researchers have managed to find the relationships between these early values and their correspondent 28-day strength. A variety of equations have been established to size up this 28-day compressive strength. Predicting with this kind of methods is more accurate. On the other hand, this still needs a period to cure and to measure the specimens, and can't provide the required experimental data to manufacture and construction sites in time.

It is well known that the long-term properties of concrete can be improved by controlling the grade of cement, water/cement ratio, amount of cement, amount of water, slump, etc., of fresh concrete within specified limits. Since the data about fresh concrete is routinely collected and has been used to provide quality control direction, it appears reasonable that this data can also be used to predict the long-term strength of concrete.

Methods of the second category are carried out as follows: these analyze the composition of concrete and their respective characteristics, sort out some factors that affect the compressive strength of concrete, and induce function equations between these factors and concrete strength by the regressive theory. These calculate concrete strength by substitution of the tested values of some parameters of freshly mixed concrete in the equations. This type of methods is simple and inexpensive, but sometimes gives results with a lower accuracy. The reason is that there are too many parameters related to compressive strength for a regressive analysis, especially due to their stochastic and fuzzy properties. Furthermore, with the development of concrete technology, many admixtures are being invented to improve properties such as strength, workability, endurance, etc. Predicting these properties is even more difficult, therefore it is beneficial to explore other new approaches.

In recent years, artificial intelligence has gradually developed as an information processing means. The study of artificial neural networks (ANN) was inspired by advances in biological neural network research and neural computing has emerged as a practical alternative to computational algorithms for its strong parallel calculation and complex nonlinear mapping abilities. The multi-layer feed-forward neural network model is one of the most commonly used ANN models, its applications extend to almost every field (Yeh, 1999).

Therefore, it is theoretically feasible to consider predicting concrete properties by using the multi-layer feed-forward neural networks (MFNNs). In this paper, a nonlinear mapping model of neural networks based upon a global optimization back-propagation algorithm (GOBPA) has been constructed to deal with this concrete strength prediction problem.

Nomenclature

L_2	Euclidean norm;
ψ: $\mathbf{X} \to \mathbf{Y}$	mapping ψ from $\mathbf{X}$ to $\mathbf{Y}$;

$\mathbf{R}^N$	N-dimensional real Euclidean space;
$[0,1]^N$	N-dimensional closed domain from 0 to 1;
N_i,N_h,N_o	number of the neurons in input, hidden, and output layers respectively;
o_{pj}	estimated output of node j by a network under the pth pattern;
net_{pj}	total input of node j under the pth pattern;
θ_{pj}	threshold for neuron j in output layers under the pth pattern;
N_p	number of vectors in training set;
t_{pj}	target output of node j under the pth pattern ;
λ	variable parameter decreasing from 1 to 0;
$\forall \mathbf{X}_1,\ \mathbf{X}_2 \in \mathbf{D}$	for all $\mathbf{X}_1$, $\mathbf{X}_2$ contained in set **D**;
$\forall \beta \in (0,1)$	for all β contained in a range of 0 to 1;
E_p	total feed-forward calculation error corresponding to the pth pattern;
τ	matrix transpose;
w_{hi}	weight from node i in intput layer to node h in hidden layer ;
q	number of iteration;
η	learning rate, and $0 < \eta < 1$;
α	momentum constant, $0 < \alpha < 1$.

2. Architecture and Algorithm of the BP Neural Networks

Fig. 1 shows a typical multi-layer feed-forward neural network for prediction of concrete, which has three layers: an input layer i, a hidden layer h and an output layer j. In the input layer, nodes 1 to 11 correspond to the grade of cement, water/cement ratio, amount of water, amount of cement, maximum diameter of gravel, fine module of

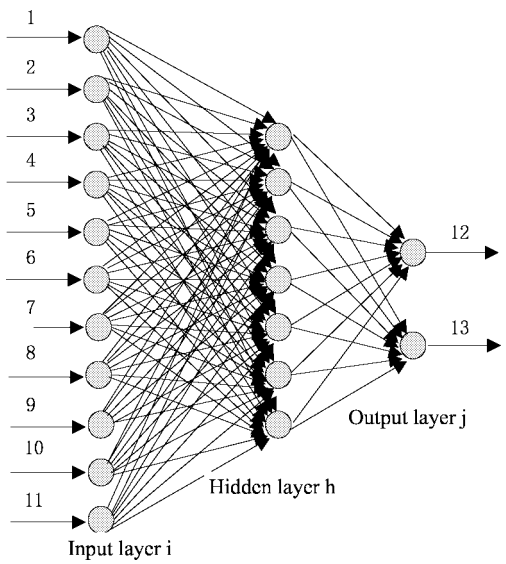

Fig. 1. Neural network model for prediction of strength of concrete.

sand, sand/aggregate ratio, aggregate/cement ratio, slump, action of admixtures, amount of admixtures. In the output layer, nodes 12 and 13 relate to the 7-day strength and 28-day strength respectively. Neurons in the neighboring layers are fully interconnected by their respective specified weights. A multi-layer feed-forward neural network with the error back-propagation algorithm is simply called a BP neural network. The BP neural networks can accomplish complicated nonlinear mapping and many complex function relationships. It does this by compounding simple nonlinear functions, such as sigmoid functions, only several times (Eberhart *et al.*, 1996).

Regarding a neural network as a mapping tool from inputs to outputs, the mapping will be highly nonlinear. Let N be the number of the input neurons, and M be the number of the output neurons, $\mathbf{R}^N$ and $\mathbf{R}^M$ be N-dimensional and M-dimensional real Euclidean space. The expression $[0,1]^N$ is a closed unit domain with N dimensions belonged to N-dimensional real space $\mathbf{R}^N$. From the works of Kolmogorov and Hecht–Nielsen, etc. the neural network can implement a mapping from $\mathbf{R}^N$ to $\mathbf{R}^M$, the mapping function is a L_2 norm function denoted by ψ. Hecht–Nielsen proved that any a L_2 norm function like

$$\psi\colon\ [0,1]^N \subset \mathbf{R}^N \to \mathbf{R}^M$$

can be approximated by a three-layer back-propagation neural network to any desired degree of accuracy in the mean squared error sense, if each of the ψ's coordinate functions is square-integrable (Hecht–Nielsen, 1991). That is to say, a three-layer BP neural network could implement any function defined over a compact subset of Euclidean space within any mean squared error accuracy.

The BP neural networks are capable of learning from example. Through learning from example, the BP networks attain an alogical induction by adjusting their connecting weights and thresholds. The implementation of the BP algorithm consists of two passes. In the first pass, the calculation of data flows forward from the input layer to the output layer. The second pass involves propagating error signals backward from the output layer to the input layer. These two passes take place alternately. The steepest descent strategy is employed to search for a set of weight vectors to minimize the error function in the weight vector space to accomplish a process of information extracting and memorizing.

Suppose sign N_i is a number of the neurons in the input layer, N_h the hidden layer, and N_o the output layer. For the input layer, each output value equals its corresponding input value. This means that the input layer simply fans out the input data. Corresponding to the pth pattern of the samples ($p = 1, 2, \ldots, N_p$), denote the estimated output of node h by o_{ph}, the total input by $\mathrm{net}_{\mathrm{ph}}$, and the threshold by θ_{ph} for the hidden layer. And the relevant symbols for other layers can be denoted similarly to those of the hidden. Many relationship equations between the inputs and outputs can be found in many references on the BP neural networks.

Define the total net input of the node h of the hidden layer and the node j of the output

one by

$$\begin{aligned} \mathrm{net_{ph}} &= \sum_{\mathrm{i=1}}^{\mathrm{Ni}} \mathrm{w_{hi} o_{pi}} - \theta_{\mathrm{ph}}, \\ \mathrm{net_{pj}} &= \sum_{\mathrm{h=1}}^{\mathrm{Nh}} \mathrm{w_{jh} o_{ph}} - \theta_{\mathrm{pj}}, \end{aligned} \tag{1}$$

where w_{hi} is a connection weight from the ith neuron in a front layer to the hth neuron in the posterior neighboring layer, specifically from the input to the hidden layer. Similarly w_{jh} is a connective weight from the hidden layer to the output one.

Neural nodes either in the hidden or output layer can perform a transformation with a mapping function from its total net input $\mathrm{net_{ph}}$ or $\mathrm{net_{pj}}$ to the relevant output o_{ph} or o_{pj}. If a sigmoid function is taken into account and denoted by f, their respective outputs of the nodes in these two layers can be expressed by

$$\begin{aligned} o_{ph} &= f(\mathrm{net_{ph}}) = \frac{1}{1+\exp(-\mathrm{net_{ph}})}, \\ o_{pj} &= f(\mathrm{net_{pj}}) = \frac{1}{1+\exp(-\mathrm{net_{pj}})}. \end{aligned} \tag{2}$$

Related to all the output nodes and all the training patterns, a sum-squared error function of the neural network can be defined by

$$E = \sum_{p=1}^{N_{\mathrm{p}}} E_p = \frac{1}{2} \sum_{p=1}^{N_p} \sum_{j=1}^{No} (t_{pj} - o_{pj})^2, \tag{3}$$

where N_p is a number of the vectors in the training set, t_{pj} is the target output (or actual value of strength) of node j in the output layer, and o_{pj} the computational output by the network.

Define error signals of node j in the output layer and node h in the hidden by

$$\delta_{pj} = -\frac{\partial E_p}{\partial \mathrm{net_{pj}}} \quad \text{and} \quad \delta_{ph} = -\frac{\partial E_p}{\partial \mathrm{net_{ph}}}. \tag{4}$$

Substituting (3) into the above, and then finding the derivative of E_p to $\mathrm{net_{pj}}$ and $\mathrm{net_{ph}}$ on a consideration of (2) and (1), the error signal of the output node will be given below:

$$\delta_{pj} = (t_{pj} - o_{pj}) \cdot f(\mathrm{net_{pj}}), \quad \mathrm{j} = 1, 2, \ldots, \mathrm{N_o}. \tag{5}$$

As the errors in the output layer propagate backward, the error signal of the hidden node will be expressed as

$$\delta_{ph} = f(\mathrm{net_{ph}}) \sum_{\mathrm{j=1}}^{\mathrm{No}} \delta_{\mathrm{pj}} \mathrm{w_{jh}} \quad \mathrm{h} = 1, 2, \ldots, \mathrm{N_h}. \tag{6}$$

Define adjustments of the connective weights between every two neighboring layers as

$$\Delta w_{jh} = -\eta \frac{\partial E_p}{\partial w_{jh}}, \quad \Delta w_{hi} = -\eta \frac{\partial E_p}{\partial w_{hi}}. \tag{7}$$

Finding the gradients of the error function related to the pth input pattern in (3), then substituting (4), (1) and the derivative of $\mathrm{net_{pj}}$ into it, we can get

$$\begin{aligned} \Delta w_{jh} &= -\eta \frac{\partial E_p}{\partial \mathrm{net_{pj}}} \cdot \frac{\partial \mathrm{net_{pj}}}{\partial w_{jh}} = \delta_{pj} o_{ph}, \\ \Delta w_{hi} &= -\eta \frac{\partial E_p}{\partial \mathrm{net_{ph}}} \cdot \frac{\partial \mathrm{net_{ph}}}{\partial w_{jh}} = \delta_{ph} o_{pi}. \end{aligned} \tag{8}$$

Then updating processes of these weights can be performed by

$$w_{jh}(q+1) = w_{jh}(q) + \Delta w_{jh}(q+1), \quad \Delta w_{jh}(q+1) = -\eta \frac{\partial Ep}{\partial w_{jh}} + \alpha \Delta w_{jh}(q), \tag{9}$$

$$w_{hi}(q+1) = w_{hi}(q) + \Delta w_{hi}(q+1), \quad \Delta w_{hi}(q+1) = -\eta \frac{\partial Ep}{\partial w_{hi}} + \alpha \Delta w_{hi}(q), \tag{10}$$

where q is an iteration number in the training pass, η is a learning rate, and $0 < \eta < 1$, and α is a momentum constant, and $0 < \alpha < 1$.

3. Global Optimization Back-Propagation Algorithm

It can be found that the sum-squared error function does not satisfy the conditions of positive definite quadratic form over the weight space by a further study of the classical or basic BP algorithm. This means that the function is not a convex quadratic form as defined in the space. There are many local minima in the hypersurface of the error function. When a learning process enters at a local minimum, the convergence of the BP algorithm is rather slow, and the network cannot reduce the error to a required accuracy.

A novel grand total error function that satisfies the conditions of positive definite quadratic form should be proposed in order to completely tackle the problem of local minima (Karayiannis, 1992; Wang and Wang, 1996). Therefore, we introduce a global optimization back-propagation algorithm, shortly called GOBPA in the output space instead of the weight space. A novel grand total error function from the feed-forward computation of neural network can be defined over the output space by

$$E = \sum_{p=1}^{N_p} \sum_{j=1}^{N_o} \left[t_{pj} \left(t_{pj} - o_{pj} \right) + \frac{1}{2} \lambda \left(o_{pj}^2 - t_{pj}^2 \right) + \frac{1}{2} \left(1 - \frac{o_{pj}}{t_{pj}} \right)^2 \right], \tag{11}$$

where λ is a variable parameter decreasing from 1 to 0 during the training period and the other parameters are the same as before.

DEFINITION 1. Supposing a convex set $\mathbf{D} \subseteq \mathbf{R}^N$, under the assumption of $\forall \mathbf{X}_1,\ \mathbf{X}_2 \in \mathbf{D}$ and $\forall \beta \in (0,1)$, if there exists

$$F(\beta \mathbf{X}_1 + (1-\beta)\mathbf{X}_2) < \beta F(\mathbf{X}_1) + (1-\beta)F(\mathbf{X}_2), \tag{12}$$

then we state that $F(\mathbf{X})$ is a strict convex function defined over $\mathbf{D}$.

DEFINITION 2. A nonlinear programming problem with inequality constraints may be expressed in mathematical terms as

$$\begin{cases} \text{minimize } F(\mathbf{X}) \\ \text{subject to } g_i(\mathbf{X}) \geqslant 0, \quad i = 1, 2, \ldots, m. \end{cases} \tag{13}$$

In the expression (13), minimum of the function $F(\mathbf{X})$ is required, subjecting to a set of inequality constraints, which may differ from each other. Here $\mathbf{X}$ is an N-dimensional column vector, in which each component is a decision variable, and $g_i(\mathbf{X})$ is the ith inequality constraint. Assuming that $F(\mathbf{X})$ and $-g_i(\mathbf{X})$ $(i = 1, 2, \ldots, m)$ are strict convex functions, then we say the nonlinear constrained program (13) is a convex one.

Theorem 1. *Assuming $F(\boldsymbol{X})$ is a strict convex function, $-g_i(\boldsymbol{X})$ $(i = 1, 2, \ldots, m)$ are convex functions too, and assuming the optimum set $\mathbf{R}_{opti} \neq \boldsymbol{\Phi}$, where $\boldsymbol{\Phi}$ means an empty set, then there exists only one solution to the program in the expression* (13) *(Chen, 1985).*

Theorem 2. *The grand total error function E as represented by* (11) *is a strict convex function defined over the output space.*

Proof of Theorem 2. Without any specified qualification, E_p is a part of the grand total error E defined in (11), corresponding to the pth pattern (i.e., under the pth input vector), and can be deduced as follows:

$$\begin{aligned} E_p &= \sum_{j=1}^{N_O} \left[t_{pj}(t_{pj} - o_{pj}) + \frac{\lambda}{2}\left(o_{pj}^2 - t_{pj}^2\right) + \frac{1}{2}\left(1 - \frac{o_{pj}}{t_{pj}}\right)^2 \right] \\ &= \sum_{j=1}^{N_O} \left[\frac{1}{2}\left(\lambda + \frac{1}{t_{pj}^2}\right)(t_{pj} - o_{pj})^2 + (1-\lambda)\, t_{pj}\,(t_{pj} - o_{pj}) \right]. \end{aligned} \tag{14}$$

Define some vector symbols below

$$\mathbf{O}^\tau = [o_{p1},\ o_{p2},\ \ldots,\ o_{pNo}], \tag{15}$$

$$\mathbf{T}^\tau = [t_{p1},\ t_{p2},\ \ldots, t_{pNo}], \tag{16}$$

$$\mathbf{X}^\tau = (\mathbf{T} - \mathbf{O})^\tau = [t_{p1} - o_{p1},\ t_{p2} - o_{p2},\ \ldots,\ t_{pNo} - o_{pNo}], \tag{17}$$

$$\mathbf{A} = \begin{bmatrix} \lambda + \frac{1}{t_{p1}^2} & 0 & \cdots & 0 \\ 0 & \lambda + \frac{1}{t_{p2}^2} & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & \lambda + \frac{1}{t_{pN_O}^2} \end{bmatrix}, \tag{18}$$

where the superscript τ is a sign of matrix transpose. Substituting equations (15), (16), (17) and (18) into (14), then (14) is simplified as

$$E_p(\mathbf{X}) = \frac{1}{2}\mathbf{X}^\tau \mathbf{A}\mathbf{X} + (1-\lambda)\mathbf{T}^\tau \mathbf{X}. \tag{19}$$

Given the above, for any N_o-dimensional output space $\mathbf{Q} \subseteq \mathbf{R}^{No}$, if $\forall \mathbf{X}_1,\ \mathbf{X}_2 \in \mathbf{Q}$ and $\forall \beta \in (0,1)$ on substitution of (19), the following expression

$$E_p(\beta\mathbf{X}_1 + (1-\beta)\mathbf{X}_2) - \beta E_p(\mathbf{X}_1) - (1-\beta)E_p(\mathbf{X}_2) \tag{20}$$

can be deduced, without the detailed process, as an equation like

$$\begin{aligned} &E_p(\beta\mathbf{X}_1 + (1-\beta)\mathbf{X}_2) - \beta E_p(\mathbf{X}_1) - (1-\beta)E_p(\mathbf{X}_2) \\ &\quad = \frac{1}{2}\beta(\beta-1)(\mathbf{X}_1 - \mathbf{X}_2)^\tau \mathbf{A}(\mathbf{X}_1 - \mathbf{X}_2). \end{aligned}$$

From the Definition 1, the above conditions, and the positive definiteness of the matrix **A**, thus the value of the right side of the above equation is below zero. That is to say that

$$E_p(\beta\mathbf{X}_1 + (1-\beta)\mathbf{X}_2) < \beta E_p(\mathbf{X}_1) + (1-\beta)E_p(\mathbf{X}_2).$$

According to Definition 1, $E_p(\mathbf{X})$ is a strict convex function in the output space **Q**, and so is $E(\mathbf{X})$; because the grand total error of the network can be written in a form below

$$E(\mathbf{X}) = \sum_{p=1}^{N_p} E_p(\mathbf{X}) \tag{21}$$

This completes the proof of the Theorem 2.

From Theorems 1 and 2, we draw a conclusion that the only optimum solution can be obtained by optimizing the grand total error function E over the output space.

There are several ways to maximally decrease the error in (11). One of them is to minimize the error function directly. An applicable selection is by using an optimization toolbox in MATLAB, a kind of powerful technical computational software produced by the Mathworks Corporation. MATLAB has developed to versions 6.x and served a broad range of technical tasks.

In its companion optimization toolbox, many functions fit in with various optimal problems. “Fminuc” is an interested one for multivariate nonlinear unconstrained optimization, theoretically based on direct search methods and gradient methods, such as quasi-Newton method and conjugate gradient method. The conjugate gradient method is effective to speed convergence and can be expected to find the optimum solution in a finite number of iterations. At each iteration step one evaluates the current negative gradient vector and adds to it a linear combination of the previous direction vectors to obtain a new search direction, along which to move. The search directions are determined sequentially at every step and all of them are mutually conjugate.

Functions “fminicon” and “lsqnolin” in MATLAB 6.3 is for solving a multivariate nonlinear constrained problem. The second one, lsqnolin, based on Gauss–Newton and Levenberg–Marquardt methods, is qualified for nonlinear least-squares problems subject to simple bound constrains. In the optimization environment, one sets “options” with “Levenberg–Marquardt” on, then the $L-M$ search method runs. The $L-M$ search direction is evaluated between these following two directions educed from steepest-descent method and Gauss–Newton method respectively, controlled by assigning value of a nonnegative scalar λ (“lambda” in MATLAB parameters), so called a damping factor.

Artificial neural network is another choice to map a nonlinear relationship of divergent data from input to output. Concrete is composed of a lot of raw materials. Taking qualities and quantities of these raw materials as input data, and concrete strength as output data, decreasing the grand total error defined in (11) is a requirement for neural network’s convergence during training phase. In regard to the concerned specified case in this paper, processing the equation (11) is a middle step in order to deduce a set of formulae for forward and backward computations of neural networks. The formulae include weights’ adjustment and errors’ computation of the neurons in the various layers, shown in the following Eqs. (25) – (28). Convergence of the neural network is an iterative process along gradient descent direction of the error function in a simple output space.

Simple formulation of GOBPA is deduced as follows. Partially differentiate the error in (14) with respect to connective weights of a neuron, define an error signal δ_{pj} of neuron j in the output layer by

$$\delta_{pj} = \frac{-\frac{\partial E_p}{\partial o_{pj}}}{\sum\limits_{h=1}^{N_h} \left(\frac{\partial o_{pj}}{\partial w_{jh}}\right)^2 + \sum\limits_{i=1}^{Ni} \left(\frac{\partial o_{\mathrm{pj}}}{\partial w_{hi}}\right)^2}, \quad j = 1, 2, \ldots, N_o, \tag{22}$$

where w_{jh} is the weight between the output node j and the hidden node h and w_{hi} between the hidden node h and the input node i under the pth input pattern. Finding the partial derivative of E_p with respect to o_{pj}, from (14), the numerator of the right side of (22) will be

$$-\frac{\partial E_p}{\partial o_{pj}} = \left[\left(\lambda + 1/t_{pj}^2\right)\left(t_{pj} - o_{pj}\right) + \left(1-\lambda\right) t_{pj}\right], \tag{23}$$

and the denominator of (22) is a sum of squared derivations of the output of node j in the output layer with respect to the weight either from the hidden layer or from the input

layer. Finding derivations of o_{pj} to w_{jh} and to w_{hi} successively from (2) and (1), the following equation can be deduced as

$$\sum_{h=1}^{N_h}\left(\frac{\partial o_{pj}}{\partial w_{jh}}\right)^2+\sum_{i=1}^{Ni}\left(\frac{\partial o_{pj}}{\partial w_{hi}}\right)^2$$
$$=\left[o_{pj}(1-o_{pj})\right]^2\left[1+\sum_{h=1}^{N_h}\left[(o_{ph})^2+\left(w_{jh}o_{ph}\,(1-o_{ph})\right)^2\left[1+\sum_{i=1}^{N_i}(o_{pi})^2\right]\right]\right]. \quad (24)$$

Substituting (23) and (24) into (22), instead of (5) in the basic BP algorithm, the error of the output node j in GOBPA will be

$$\delta_{pj}=\left[\left(\lambda+1/t_{pj}^2\right)(t_{pj}-o_{pj})+(1-\lambda)\,t_{pj}\right]$$
$$/\left[o_{pj}\,(1-o_{pj})\right]^2\left[1+\sum_{h=1}^{N_h}\left[(o_{ph})^2+\left(w_{jh}o_{ph}\,(1-o_{ph})\right)^2\left[1+\sum_{i=1}^{N_i}(o_{pi})^2\right]\right]\right]. \quad (25)$$

Rather than the (6) in the basic BP algorithm, define the error signals in the hidden layer by

$$\delta_{ph}=\sum_{j=1}^{N_o}\delta_{pj}o_{pj}(1-o_{pj})w_{jh},\quad h=1,2,\ldots,N_h. \quad (26)$$

Substituting the (27) – (28) for the (8) – (10), adjustments of the weights between the neurons in the hidden layer and the output layer in the GOBPA are performed by

$$\Delta w_{jh}(q+1)=\eta\sum_{p=1}^{N_p}\left(\delta_{pj}o_{pj}\,(1-o_{pj})\,o_{ph}\right)+\alpha\Delta w_{jh}(q),$$
$$w_{jh}(q+1)=w_{jh}(q)+\Delta w_{jh}(q+1),\quad j=1,2,\ldots,N_o,\ h=1,2,\ldots,N_h, \quad (27)$$

where q is the iteration number in the training period. Adjust weights between the input layer and the hidden layer by

$$\Delta w_{hi}(q+1)=\eta\sum_{p=1}^{N_p}\left(\delta_{ph}o_{ph}\,(1-o_{ph})\,o_{pi}\right)+\alpha\Delta w_{hi}(q),$$
$$w_{hi}(q+1)=w_{hi}(q)+\Delta w_{hi}(q+1),\quad h=1,2,\ldots,N_h,\ i=1,2,\ldots,N_i. \quad (28)$$

Since convergence of the neural network during training phase is achieved in the output space, in which the error function (11) has a convex property, GOBPA can make a stochastic initial point converge to the global optimum solution by iterations. All of the algorithms concerning the basic BP algorithms and the GOBPA have been programmed in the C^{++} language by the authors. Further, these C^{++} source code programs have been used on computers to implement the multi-layer feed-forward neural network with these various algorithms.

4. Prediction of Concrete Strength and Comparison of the Algorithms

The prediction method suggested here belongs to the second category mentioned in the introduction of this paper. Engineers in our laboratory made a great deal of experiments on concrete. They prepared a variety of concrete specimens, cured them under various conditions at different periods and then tested their respective cubic compressive strengths (Ni and Wang, 2000). The experimental data amounts to 65 sets in all, and is divided into two parts: a large part (55 sets) is used in training neural networks called the training set, and a small part (ten sets) which is used in testing performance or accuracy of the networks' prediction called test set. Each set consists of numerous vectors of the influencing factors and the corresponding concrete strength.

Pick out fourteen test parameters as the factors (variables) that either decide or influence concrete strength (Wang *et al.*, 1999). These factors are the grade of cement, water/cement ratio, amount of water and cement, the maximum diameter of the gravel, fine module of sand, sand/aggregate ratio, aggregate/cement ratio, slump, the action of admixtures, amount of admixtures, forming conditions, curing conditions and test conditions. Of all the above factors that are obtained during a certain period of time, the following three, forming conditions, curing conditions and testing conditions, can be seen as constant factors. Therefore, the remaining eleven factors are decisive in developing the strength of concrete.

Given the above, the neural network has a structure of 11–7–2. There are eleven nodes in the input layer, and seven in the hidden layer, and two (corresponding to the 7-day and 28-day concrete strength respectively) in the output layer. The initial weights are generated at random from 0 to 1. The activation function we used is a sigmoid function defined in (2).

First the basic BP algorithm was applied. Further**,** the whole training process on the training set was implemented on computers. The sum-squared error was 0.05315, which is obviously a local minimum because the error value did not decrease much from the starting value of 0.1. Tests of the learning effect were carried out on the test set to see if their recall and generalization abilities were powerful enough. From the test phase, results estimated by the neural network were not adequate. The maximum absolute values of the test errors were 6.86MPa and the maximum absolute values of relative errors were 13.2%. Some improvements on the basic BP algorithm were also employed by the authors, such as the variation of learning rate, the variation of activation functions and simulated annealing. Of those, the strategy of varying learning rates produced a relatively good learning result with a sum-squared error of 0.04097. That showed that the improved BP algorithms still did not throw off the local minimum. In the end, the authors employed the GOBPA and then obtained a satisfactory learning result: taking only 1,000 iterations, sum-squared error being 0.00046. Tests of the learning effect were carried out on the test set. The maximum absolute value of test errors is 2.80Mpa, and the maximum absolute value of relative errors is 5.52%, which is much smaller than the error given by the basic BP algorithm. This showed the learning phase by the application of the GOBPA has been successfully completed. Table 1 gives the simulation results by five different algorithms,

Table 1
Comparison of computer simulated results

No.	Algorithms	Training phase		Testing phase	
		Number of iteration	Training error	Maximal |error| (Mpa)	Maximal |relative error| (%)
1	Basic BP algorithm	50000	0.05315	6.86	13.2
2	Varying learning rate	40000	0.04097	5.43	11.6
3	Varying activation function	40000	0.05091	5.81	12.3
4	Simulated annealing	30000	0.04413	5.29	12.7
5	GOBPA	1000	0.00046	2.80	5.52

including their iteration numbers and grand total errors at convergence during the training phase, and the maximum absolute and relative errors during the testing phase. Among them the GOBPA produced the best results with the least errors, avoiding local minima.

Variation of the errors and their convergences of the above mentioned algorithms with the number of iterations during the training are shown in Fig. 2. The vertical axis is the grand total training error, and the horizontal axis is the iteration numbers during the training sweep. Curve 1 is for the basic BP algorithm, curve 2 for variation of the activation function, curve 3 for simulated annealing, curve 4 for variation of the learning rate, and the fifth for the GOBPA. It can be found that the GOBPA algorithm made the grand total error to converge very quickly during the training sweep, the others were slower by contrast. In a word, the GOBPA showed an excellent ability to escape from local minima.

It is rather difficult to construct a nonlinear regressive equation to induce such a set of the data with multi-variable and strong nonlinear characteristics in concrete engineering. Computation with a linear regression is less complex, and it is better to compare the above neural models with a linear regression analysis. In order to simplify calculations, let us select less influence parameters and add another new factor, the 1-day strength

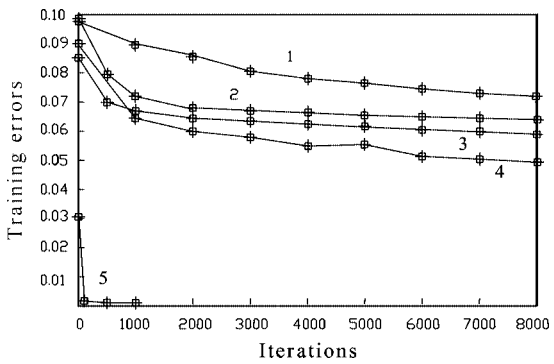

Fig. 2. Error variation curves during training.

of concrete, as independent variables of the regressive equation. It can be assumed that there exists a linear relationship between four independent variables: X_1 (the grade of cement), X_2 (the amount of water), X_3 (the cement/water ratio), and X_4 (the 1-day quick cured concrete strength) and the dependent variable $\hat{y}$ (the 28-day strength of concrete). Performing linear regressive analysis to induce this part of the data from the same experiments as mentioned above, a linear regression equation can be obtained as follows:

$$\hat{y} = -78.732 + 1.451X_1 + 0.096X_2 - 3.680X_3 + 0.414X_4. \tag{29}$$

This equation was used to predict the concrete strength on the test set, giving a maximum absolute value of forecast error of 7.8MPa, and a maximum absolute value of relative error of 19.2%. If the regressive equation did not include the independent variable X_4 for the 1-day strength, the accuracy of the prediction results was even lower from the authors' experience.

Figs. 3–5 are the histograms of test error distribution given by the BP algorithm varying the learning rate, the GOBPA and the regression analysis respectively. By comparison, the results by the BP algorithm varying the learning rate and the GOBPA are basically normally distributed in accordance with some practical experience, whereas the results by the regression form a negative skew histogram.

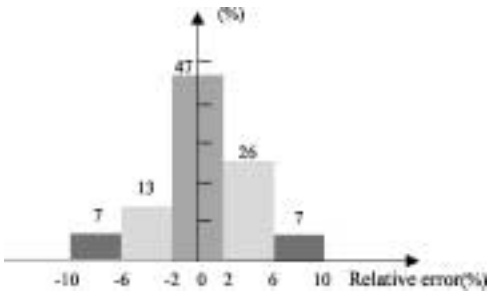

Fig. 3. Relative test errors by the variation of the learning rate.

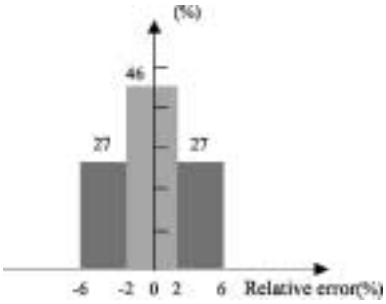

Fig. 4. Relative test errors by the GOBPA.

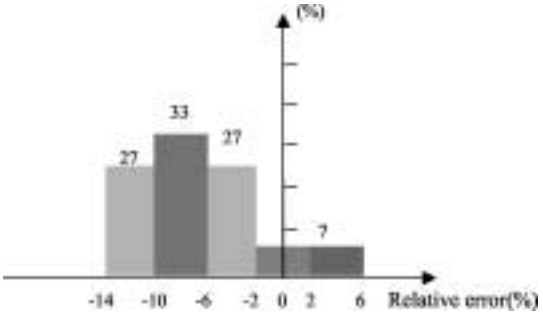

Fig. 5. Relative test errors by the regression analysis.

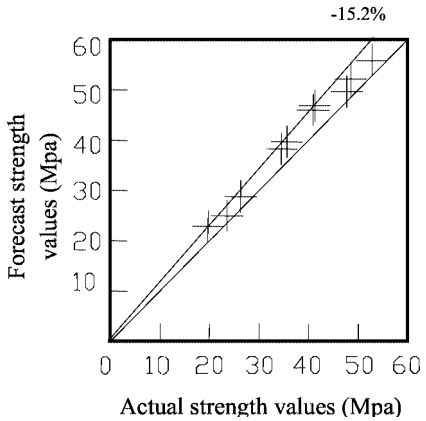

Fig. 6. Comparison of the actual values to the forecast values from the linear regression.

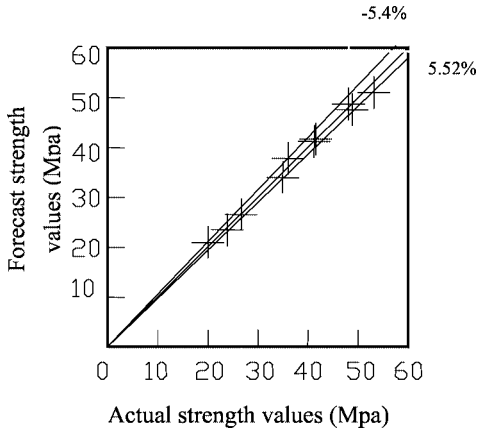

Fig. 7. Comparison of the actual values to the forecast ones from the GOBPA.

Figs. 6 and 7 are contrasts between the forecast values and the actual values of the concrete strength (selecting ten points). The horizontal axis shows the actual strength values of the experimental data and the vertical axis shows the forecast strength values from the calculation results. The Fig. 6 represents the linear regression result, and Fig. 7 is for the GOBPA neural network. The closer the points are to the diagonal, the better the results. If the actual and forecast values of a pair of data are equal, the point will fall onto the diagonal. It can be seen that the points in Fig. 7 distribute more intensely about the diagonal than those in Fig. 6, the GOBPA giving better test results.

5. Conclusions

The computation results and the comparisons of all these methods above show that the BP algorithm based on global optimization can reduce the training time of neural networks and the grand total error and much increase prediction reliability. Therefore it is an improvement on the basic BP algorithm. Application of this approach can not only simplify the concrete strength experiments in laboratories, manufactures and construction sites, but can also supply more correct and prompt data for structural design and construction control. We believe a further study of ANN will simplify concrete test work, and provide more accurate strength data for structural designs, and furthermore bring considerable economic benefits.

6. Acknowledgement

The research reported here is supported by the National Natural Science Foundation of China (No. 50178028). The authors would like to express their great thanks to them for finance grant.

References

Chen, K. Zh. (1985). *The Optimizing Computation Methods*, the Northwestern Telecommunication Institute Press, Xian, China. pp. 21–33 (in Chinese).

Eberhart, R., S. Pat and R. Dobbins (1996). *Computational Intelligence PC Tools*, Academic Press, New York.

Hecht–Nielsen, R. (1991). Theory of the back-propagation neural network. In Harry Wechsler (Eds.), *Neural Networks for Perception*, Vol. 2, Academic Press, San Diego. pp. 65–93.

Karayiannis, B. (1992). Recent development in supervised learning. In *IEEE Transaction on System, Man, and Cybernetics*, SMC, Vol. 1. pp. 387–391.

Ni, H.-G., and J.-Z. Wang (2000). Prediction of compressive strength of concrete by neural networks. *Cement and Concrete Research*, **30**(8), 1245–1250.

Wang, D.-H., and Z.-Y. Chen (1997). On prediction of compressive strength of mortars with ternary blends of cement, GGBFS and fly ash, *Cement and Concrete Research*, **27**(4), 487–493.

Wang, J.-Z., H.-G. Ni and J.-Y. He (1999). The application of automatic acquisition of knowledge to mix design of concrete, *Cement and Concrete Research*, **29**(12), 1875–1880.

Wang, K.-J., and K.-Ch. Wang (1996). *Modeling, Prediction and Control of Neural Networks*, Harbin Engineering University Press, Harbin, China. pp. 76–85 (in Chinese).

Wu, Z.-W., and H.-Z. Lian (1999). *High-Performance Concrete*, The Chinese Railroad Press, Beijing (in Chinese).

Yeh I.-C. (1999). Design of high-performance concrete mixture using neural networks and nonlinear programming, *Journal of Computing in Civil Engineering*, **13**(1), 36–42.

J.-Z. Wang born in 1943, Professor, interesting in mathematics, mechanics, artificial intelligence. He published dozens of papers indexed by SCI, EI, CA, etc.

X.-J. Wang born in 1971, engineer, interesting in civil engineering, now study in Institute for Reinforced and Prestressed Concrete Structures Ruhr-University Bochum, Germany.

H.-G. Ni born in 1973, engineer, interesting in civil engineering.

Neurotinklų algoritmas ir jo taikymas betono konstrukcijų inžinerijos duomenų apdorojime

Ji-Zong WANG, Xi-Juan WANG, Hong-Guang NI

Prognozuoti betono mechanines savybes yra sudėtinga netiesinė problema. Dirbtiniai neurotinklai buvo panaudoti siekiant išvengti tradicinių algoritmų trūkumų išskiriant domenų apdorojimo taisykles. Autoriai siūlo naują iškilią bendros paklaidos funkciją, pritaikomą globalios optimizacijos atbulo sklidimo algoritmui. Skaičiavimo rezultatai, modeliuojant betono stiprumą pasiūlytu algoritmu, buvo geresni negu taikant tiesinės regresijos analizę.