16

Confidence Intervals for Neural Networks and Applications to Modeling Engineering Materials

Shouling He[1] and Jiang Li[2]
[1]*Department of Engineering Technology, University of Pittsburgh at Johnstown PA 15904,*
[2]*Department of Civil Engineering, Morgan State University, Baltimore MD 21251,*
USA

1. Introduction

Feedforward neural networks have been theoretically proved to be able to approximate a nonlinear function to any degree of accuracy as long as enough nodes exist in the hidden layer(s) (Hornik et. al. 1989). However, when feedforward neural networks are applied to modeling physical systems in the real world, people care more about their prediction capabilities than accurate modeling abilities. If a neural network is trained with noisy data measured from an experiment, what is the predictive performance of the neural network when unseen input data is fed into it? In this chapter, the confidence interval and prediction interval of a neural network model will be discussed. In particular, how the nonlinear structure of a feedforward neural network, impacts the confidence interval will be analyzed. Then, as an application, the measure of confidence to estimate nonlinear elastic behavior of reinforced soil is demonstrated.

This chapter starts with a description of the structure of feedforward neural networks and basic learning algorithms. Then, nonlinear regression and its implementation within the nonlinear structure like a feedforward neural network will be discussed. The presentation will show confidence intervals and prediction intervals as well as applying them to a one-hidden-layer feedforward neural network with one, two or more hidden node(s). Next, it is proceeded to apply the concepts of confidence intervals to solving a practical problem, prediction of the constitutive parameters of reinforced soil that is considered as composite material mixed with soil, geofiber and lime powder. Prediction intervals for the practical case is examined so that more quality information on the performance of reinforced soil for better decision-making and continuous improvement of construction material designs can be provided. Finally, the neural network-based parameter sensitivities will be analyzed.

In order to clearly present the algorithms discussed in this chapter, some notations are declared as follows: matrices and vectors are written in boldface letters, and scalars in italics. Vectors are defined in column vectors. The superscript T of a matrix (or vector) denotes the transpose of the matrix (or vector).

2. Neural network architecture and learning algorithms

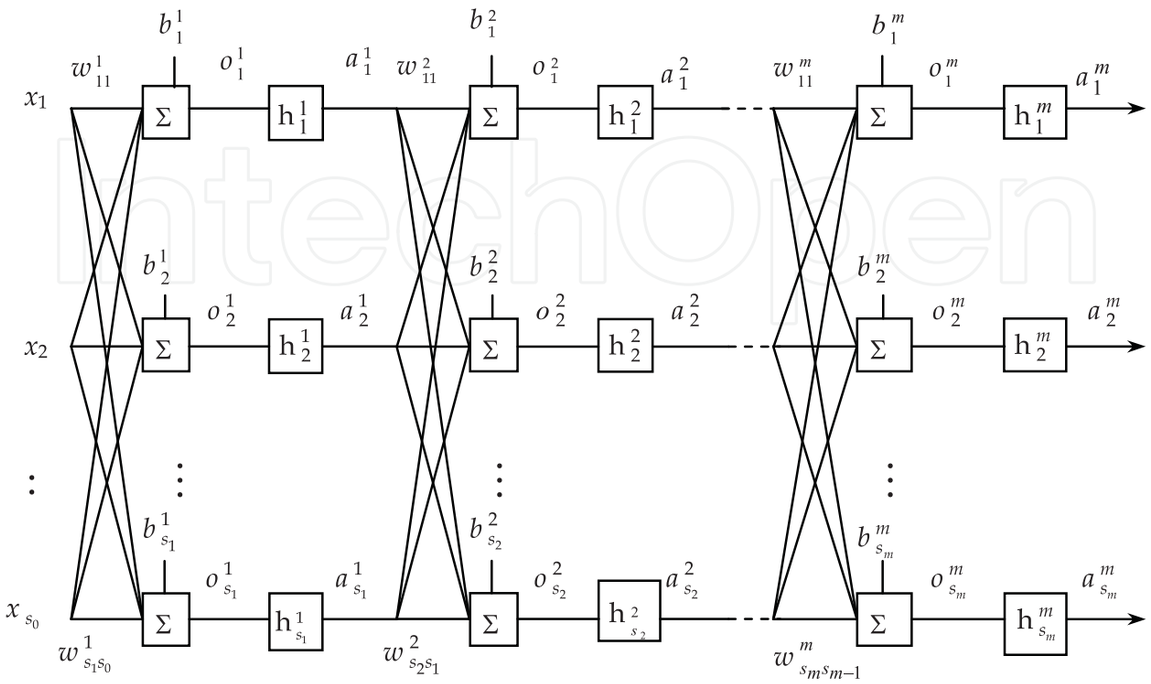


Fig. 1.1a. An m-layer feedforward neural network

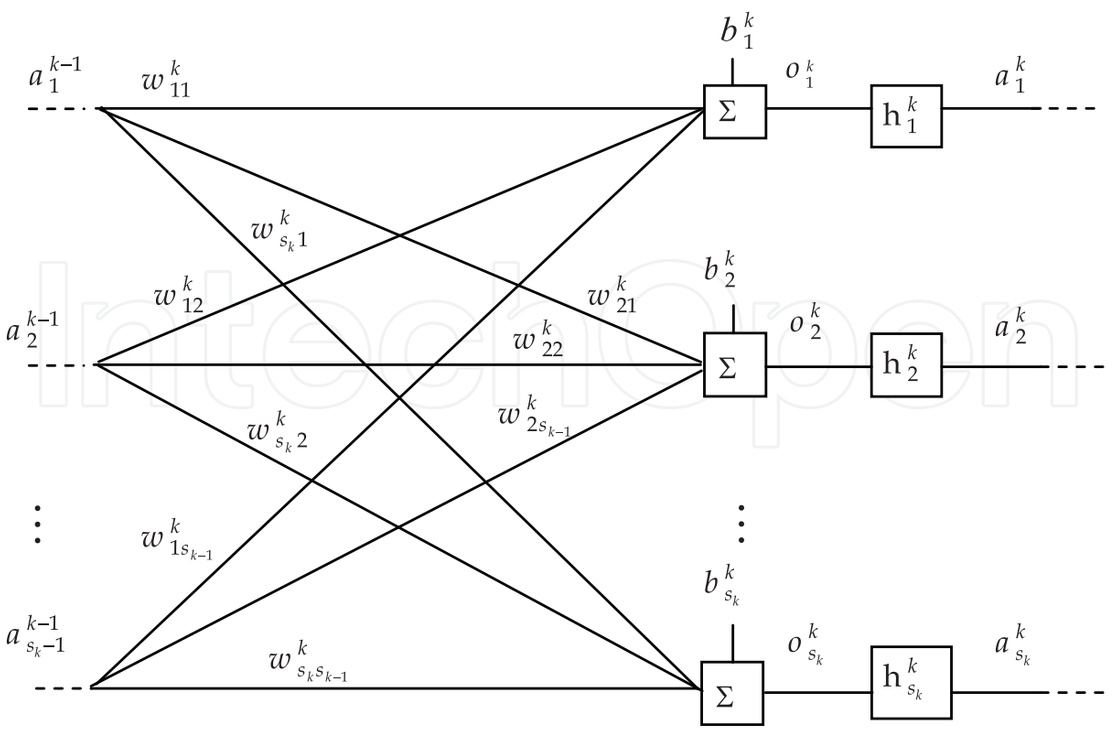


Fig. 1.1b. Weights and biases in the k^{th} layer

2.1 Architecture of feedforward neural networks

A feedforward neural network is a massive net consisting of a number of similar computing units, which are called nodes. The morphology of a neural network can change depending on the way the nodes are interconnected and the operations performed at each node. As shown in Figs. 1.1a and 1.1b, in an m-layer feedforward neural network, the nodes are arranged in layers. All nodes in a layer are fully connected to the nodes in adjacent layers by weights, adjustable parameters to represent the strength of connections. The summation of weighted inputs to a node will be mapped by a nonlinear activation function, h[.]. There are no connections between nodes in the same layer. Data information is passed through the network in such a manner that the outputs of the nodes in the first layer become the inputs of the nodes in the second layer and so on.

Mathematically, an m-layer feedforward neural network can be expressed as follows,

$$\mathbf{o}^k = \mathbf{w}^k \mathbf{a}^{k-1} + \mathbf{b}^k \quad \text{and} \quad \mathbf{a}^k = \mathbf{h}^k(\mathbf{o}^k) \qquad (k = 1, \cdots, m) \tag{1}$$

where $\mathbf{a}^0 = \mathbf{x} = [x_1 \quad \cdots \quad x_{s_0}]^{\mathrm{T}}$ is the input vector; $\mathbf{o}^k = [o_1^k \quad \cdots \quad o_{s_k}^k]^{\mathrm{T}}$, $\mathbf{h}^k = [h_1^k \quad \cdots \quad h_{s_k}^k]^{\mathrm{T}}$ and $\mathbf{a}^k = [a_1^k \quad \cdots \quad a_{s_k}^k]^{\mathrm{T}}$ are the linear output vector of the summation, the activation function vector and the output vector in the k^{th} layer, respectively; s_k is the number of nodes in the k^{th} layer; $\mathbf{w}^k$ and $\mathbf{b}^k$ represent the weight matrix and the bias vector in the k^{th} layer (see Fig. 1.1b), which can be respectively expressed by

$$\mathbf{w}^k = \begin{bmatrix} w_{11}^k & \cdots & w_{1s_{k-1}}^k \\ \vdots & \ddots & \vdots \\ w_{s_k 1}^k & \cdots & w_{s_k s_{k-1}}^k \end{bmatrix} \quad \text{and} \quad \mathbf{b}^k = \begin{bmatrix} b_1^k \\ \vdots \\ b_{s_k}^k \end{bmatrix}, \tag{2}$$

in which the j^{th} row of $\mathbf{w}^k$ is defined by $\mathbf{w}_j^k = [w_{j1}^k \quad w_{j2}^k \quad \cdots \quad w_{js_{k-1}}^k]$ (j=1, …, s_k).

2.2 Learning algorithms

2.2.1 Standard backpropagation

Given a set of s_0-dimensional input vector, $\mathbf{x}_i$, $_{(i=1,\ldots,Q)}$, and the corresponding s_m-dimensional output vector, $\mathbf{t}_i$, $_{(i=1,\ldots,Q)}$, the weights and biases of a feedforward neural network are adjusted such that the following performance index is minimum,

$$E = \sum_{i=1}^{Q} E_i \quad \text{with} \quad E_i = \frac{1}{2}(\mathbf{t}_i - \mathbf{a}_i^m)^{\mathrm{T}}(\mathbf{t}_i - \mathbf{a}_i^m) \tag{3}$$

where $\mathbf{a}_i^m = \mathbf{a}_i^m(\mathbf{x}_i)$ is the output of the feedforward neural network with input $\mathbf{x}_i$ and Q is the number of samples. Since the structure of a feedforward neural network is the same for all samples, for simplicity, the subscript i will be dropped in the derivation of the backpropagation algorithm.

For a single input/output sample, Equation (3) is denoted by E_i. According to the gradient descent algorithm, the weight matrix and bias vector of the k^{th} layer will be updated according to the following equations so that E_i can be minimized,

$$\Delta \mathbf{w}^k = -\eta(\partial E_i / \partial \mathbf{w}^k), \qquad \Delta \mathbf{b}^k = -\eta(\partial E_i / \partial \mathbf{b}^k)^{\mathrm{T}} \tag{4}$$

where η is the learning rate ($\eta > 0$).

By defining the gradient of E_i with respect to the linear output vector $\mathbf{o}^k$ of the k^{th} layer as

$$\boldsymbol{\delta}^k := \nabla_{\mathbf{o}^k} E_i = [\,\delta_1^k \quad \delta_2^k \quad \cdots \quad \delta_{s_k}^k\,]^{\mathrm{T}}, \qquad (k = 1,\cdots,m), \tag{5}$$

the differentiation of E_i with respect to the weight matrix and bias vector is presented as follows, (See Appendix for application of the chain rule to the differentiation of a scalar function with respect to a matrix.)

$$\frac{\partial E_i}{\partial \mathbf{w}^k} = \sum_{j=1}^{s_k} \frac{\partial E_i}{\partial o_j^k}\frac{\partial o_j^k}{\partial \mathbf{w}^k} = \delta_1^k \begin{bmatrix} (\mathbf{a}^{k-1})^{\mathrm{T}} \\ \mathbf{0}_{(n-1)\times n} \end{bmatrix} + \cdots + \delta_{s_k}^k \begin{bmatrix} \mathbf{0}_{(n-1)\times n} \\ (\mathbf{a}^{k-1})^{\mathrm{T}} \end{bmatrix} = \boldsymbol{\delta}^k \cdot (\mathbf{a}^{k-1})^{\mathrm{T}}, \tag{6}$$

$$\frac{\partial E_i}{\partial \mathbf{b}^k} = \frac{\partial E_i}{\partial \mathbf{o}^k}\frac{\partial \mathbf{o}^k}{\partial \mathbf{b}^k} = (\boldsymbol{\delta}^k)^{\mathrm{T}},$$

where $o_j^k = \mathbf{w}_j^k \mathbf{a}^{k-1} + b_j^k \;\; (j = 1,\cdots,s_k)$.

From Equations (1) and (3), it can be seen that E_i is a function of the vector $\mathbf{o}^{k+1}$ and $\mathbf{a}^k$ is also a function of the vector $\mathbf{o}^k$. Using the general chain rule (See Appendix), therefore, it leads to the following relation,

$$\partial E_i / \partial \mathbf{o}^k = (\partial E_i / \partial \mathbf{o}^{k+1})(\partial \mathbf{o}^{k+1} / \partial \mathbf{a}^k)(\partial \mathbf{a}^k / \partial \mathbf{o}^k). \tag{7}$$

Again, by applying the general chain rule and the definition (5) of $\boldsymbol{\delta}^k$, the recurrence relation of the gradient term $\boldsymbol{\delta}^k$ can be written by

$$\begin{aligned} \boldsymbol{\delta}^k &= \nabla_{\mathbf{o}^k} E_i = \nabla_{\mathbf{o}^k} \mathbf{a}^k \cdot \nabla_{\mathbf{a}^k} \mathbf{o}^{k+1} \cdot \nabla_{\mathbf{o}^{k+1}} E_i \\ &= \nabla_{\mathbf{o}^k} \mathbf{a}^k \cdot \nabla_{\mathbf{a}^k} \mathbf{o}^{k+1} \cdot \boldsymbol{\delta}^{k+1} = \dot{\mathbf{H}}^k(\mathbf{o}^k) \cdot (\mathbf{w}^{k+1})^{\mathrm{T}} \cdot \boldsymbol{\delta}^{k+1}, \qquad (k = 1, \cdots, m-1), \end{aligned} \tag{8}$$

where

$$\nabla_{\mathbf{a}^k} \mathbf{o}^{k+1} = (\partial \mathbf{o}^{k+1} / \partial \mathbf{a}^k)^{\mathrm{T}} = (\mathbf{w}^{k+1})^{\mathrm{T}}; \tag{9}$$

$$\nabla_{\mathbf{o}^k} \mathbf{a}^k = (\partial \mathbf{a}^k / \partial \mathbf{o}^k)^{\mathrm{T}} = \dot{\mathbf{H}}^k(\mathbf{o}^k) = \begin{bmatrix} \dot{h}_1^k(o_1^k) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \dot{h}_{s_k}^k(o_{s_k}^k) \end{bmatrix}; \; (\dot{h}_j^k(o_j^k) = dh_j^k(o_j^k)/do_j^k, \; j=1,\ldots,s_k. \tag{10}$$

This recurrence computation is initialized at the final layer, i.e. the m^{th} layer. According to the general chain rule, $\boldsymbol{\delta}^m$ will be

$$\boldsymbol{\delta}^m = \nabla_{\mathbf{o}^m} E_i = \nabla_{\mathbf{o}^m} \mathbf{a}^m \cdot \nabla_{\mathbf{a}^m} E_i = -\dot{\mathbf{H}}^m(\mathbf{o}^m) \cdot (\mathbf{t}_i - \mathbf{a}_i^m). \tag{11}$$

The learning algorithm of the standard backpropagation proceeds as follows: first, using Equation (1) to calculate the output of each layer $\mathbf{a}^k$ (k=1,…,m); Then, using Equations (11) and (8), the gradient terms $\boldsymbol{\delta}^k$ (k=m,…,1) is computed backward from the m^{th} layer to the 1st layer; Next, the increments of weights and biases are calculated using Equations (6) for k=1,...,m; Finally, the weights and biases are updated using Equations (4) with a chosen learning rate η (k=1,..., m).

2.2.2 Levenberg-Marquardt backpropagation algorithm

The standard backpropagation algorithm has been widely applied in neural network learning. However, due to the low speed of convergence, considerable research works have been done to improve it. A lately developed algorithm, the Levenberg-Marquardt backpropagation, has been used to train feedforward neural networks since it can yield a speed-up of large factors via limited modifications of the standard backpropagation algorithm.

Consider the feedforward neural network (1) as a nonlinear least squares problem, the performance index (3) can be written as below,

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{Q}(\mathbf{t}_i - \mathbf{a}_i^m)^{\mathrm{T}}(\mathbf{t}_i - \mathbf{a}_i^m) = \frac{1}{2}(\mathbf{t}-\mathbf{a})^{\mathrm{T}}(\mathbf{t}-\mathbf{a}) \tag{12}$$

where $\mathbf{t} = [\mathbf{t}_1^{\mathrm{T}} \quad \mathbf{t}_2^{\mathrm{T}} \quad \cdots \quad \mathbf{t}_{\mathrm{Q}}^{\mathrm{T}}]^{\mathrm{T}}$ and $\mathbf{a} = [(\mathbf{a}_1^m)^{\mathrm{T}} \quad (\mathbf{a}_2^m)^{\mathrm{T}} \quad \cdots \quad (\mathbf{a}_{\mathrm{Q}}^m)^{\mathrm{T}}]^{\mathrm{T}}$ respectively.

The n-element vector of weights and biases of an m-layer neural network can be written as

$$\begin{aligned}\mathbf{w} &= [w_{11}^1 \quad w_{12}^1 \quad \cdots \quad w_{s_1}^1 \quad b_1^1 \quad \cdots \quad b_{s_1}^1 \quad w_{11}^2 \quad w_{12}^2 \quad \cdots]^{\mathrm{T}} \\ &= [w_1 \quad w_2 \quad \cdots \quad w_n]^{\mathrm{T}}\end{aligned}. \tag{13}$$

With the Newton method, the increment $\Delta\mathbf{w}$, by minimization of E with respect to the parameter vector $\mathbf{w}$, is

$$\Delta\mathbf{w} = -(\nabla^2 E(\mathbf{w}))^{-1}\nabla E(\mathbf{w}), \tag{14}$$

where $\nabla^2 E(\mathbf{w})$ is the Hessian matrix of $E(\mathbf{w})$ and $\nabla E(\mathbf{w})$ is the gradient of $E(\mathbf{w})$.

Given the performance index (12), the gradient and the Hessian matrix of $E(\mathbf{w})$ can be written as

$$\nabla E(\mathbf{w}) = -\mathbf{J}^{\mathrm{T}}(\mathbf{w})\cdot(\mathbf{t}-\mathbf{a}(\mathbf{w}))$$

$$\nabla^2 E(\mathbf{w}) = \mathbf{J}^{\mathrm{T}}(\mathbf{w})\cdot\mathbf{J}(\mathbf{w}) + \sum_{i=1}^{Q}(\mathbf{t}_i - \mathbf{a}_i^m)\nabla^2(\mathbf{t}_i - \mathbf{a}_i^m) \tag{15}$$

where $\mathbf{J}(\mathbf{w})$ is the Jacobian matrix of $\mathbf{a}(\mathbf{w})$ with respect to the vector $\mathbf{w}$,

$$\mathbf{J}(\mathbf{w}) = \partial\mathbf{a}(\mathbf{w})/\partial\mathbf{w} = \begin{bmatrix} \partial\mathbf{a}_1^m/\partial w_1 & \cdots & \partial\mathbf{a}_1^m/\partial w_n \\ \vdots & \ddots & \vdots \\ \partial\mathbf{a}_Q^m/\partial w_1 & \cdots & \partial\mathbf{a}_Q^m/\partial w_n \end{bmatrix}. \tag{16}$$

Since the second term on the right-hand side of Equation (15) is difficult to obtain, the Levenberg-Marquardt method is introduced to approximate the function as follows,

$$\Delta\mathbf{w} = -(\mathbf{J}^{\mathrm{T}}(\mathbf{w})\cdot\mathbf{J}(\mathbf{w}) + \mu\mathbf{I})^{-1}\cdot\mathbf{J}^{\mathrm{T}}(\mathbf{w})\cdot(\mathbf{t}-\mathbf{a}(\mathbf{w})) \tag{17}$$

where $\boldsymbol{I}$ is the identity matrix and μ is an adaptive factor ($\mu > 0$). μ is multiplied by a positive parameter γ (normally chosen as 10) whenever a step results in a decreased $E(\mathbf{w})$ in

Equation (12). Otherwise, μ is divided by γ. Note that when μ is sufficiently large, the algorithm becomes the steepest gradient descent. For a small value of μ, the algorithm becomes the Gauss-Newton algorithm.
In order to apply the backpropagation technique to solving the Jacobian matrix (16), the research work reported by Hagan and Menhaj (Hagan & Menhaj, 1994) provided a detailed algorithm with which the elements of the Jacobian matrix (16) can be calculated backward layer-by-layer, and therefore, the weights of a neural network can be updated simultaneously.

3. Parameter estimates with confidence intervals

The purpose to train a neural network is not solely to get an exact representation of the training data, but to build a satisfactory model that can exhibit intrinsic relationship between input data and output data. Therefore, the trained neural network model is expected to make good predictions for unseen input data. Hence, the performance evaluation of a neural network with unseen data has been intensively studied in the area of applied neural computations.
Traditionally, the performance of generalization of a neural network is examined by testing data, i.e. a set of data is separated into two subsets, training data and testing data, respectively. The neural network trained using the training data should also result in small sum of squared errors (3) when the testing data is fed into it. The method can detect whether a neural network overfits noisy data, but it does not provide quantitative metric to show "how good" the predicted output is.
On the other hand, as far as empirical modeling is concerned, regression analysis is a widely used statistical technique in many practical cases. Since a neural network can be considered as a special nonlinear structure, neural network modeling can be categorized into a nonlinear regression problem.
When a neural network model is used for prediction with a set of inputs that are different from the training patterns, the accuracy of estimation can be represented by a best guess of predicted outcomes plus a range of likely future outcomes around the best guess. Such a range is commonly referred to as a confidence interval with certain confidence level, which provides information indicating where the output is likely to be and how much percent of chances the output is probably to be with the estimates. Hence, solving the neural network regression problem consists of two parts - developing a nonlinear regression model and computing the range of likely future outputs. The range of possible outputs will quantitatively provide how large difference between the real output and the best guess from a statistical point of view when unseen data are fed into the model. Moreover, as discussed below, the confidence interval varies with the structure of a neural network, which provides a practical reference for people to select the structure of a neural network. In this section, the concepts of neural network regression, confidence intervals and prediction intervals will be presented. Then, how a prediction interval changes with the structure of a neural network will be demonstrated through an example.

3.1 Prediction interval for neural network regression

From Equation (1), it is assumed that the true output of an m-layer neural network is $\mathbf{a}_i^m(\mathbf{x}, \mathbf{w}^*)$, where $\mathbf{x}$ is the input vector, and $\mathbf{w}^*$ represents the true values of the weight vector from the weight value space $\mathbf{\Omega}$. For simplicity, $\mathbf{a}_i^m$ is replaced by $\mathbf{a}_i$ for future derivation. In

addition, the output of the neural network will be considered as a one-dimensional vector, i.e. $\mathbf{a}_i=a_i$. The error, ε_i, associated with the function in modeling is supposed to be independently and identically distributed with variance, σ^2, where the distribution has the form $N(0,\sigma^2)$, i.e. normal distribution with the mean of zero and the variance of σ^2. With each of Q experimental data, the output of the function is represented by

$$a_i = a_i(\mathbf{x}_i, \hat{\mathbf{w}}) + \varepsilon_i , \quad i=1,2,..,Q; \qquad \hat{\mathbf{w}} \in \mathbf{\Omega} \tag{18}$$

The estimated vector, $\hat{\mathbf{w}}$, is obtained by minimizing the performance index (12) using training data. However, due to many factors, e.g. noisy training patterns or limited number of nodes, the vector, $\hat{\mathbf{w}}$, can be a good estimation of, or say close to, the true value, $\mathbf{w}^*$, of the weight parameters but not equal to them. Considering a neural network as a nonlinear regression model, the linear approximation to this nonlinear regression model can be obtained via the Taylor series expansion of the function to the first order (Seber et al., 1989). Therefore, an estimated value, $\hat{a}_i$, under the input vector, $\mathbf{x}_i$, is

$$\hat{a}_i = a_i(\mathbf{x}_i, \hat{\mathbf{w}}) = a_i(\mathbf{x}_i, \mathbf{w}^*) + \nabla_{\mathbf{w}}^{\mathrm{T}} a_i \mid_{w^*} (\hat{\mathbf{w}} - \mathbf{w}^*) \tag{19}$$

where $\nabla_{\mathbf{w}}^{\mathrm{T}} a_i \mid_{w^*}$ denotes the gradient of the function a_i with respect to the weight vector at the true values, $\mathbf{w}^*$. The error between the input/output pairs and the estimated value from the neural network model yields

$$\begin{aligned} t_i - \hat{a}_i &= t_i - a_i(\mathbf{x}_i, \mathbf{w}^*) - \nabla_{\mathbf{w}}^{\mathrm{T}} a_i \mid_{w^*} (\hat{\mathbf{w}} - \mathbf{w}^*) \\ &= \varepsilon_i - \nabla_{\mathbf{w}}^{\mathrm{T}} a_i \mid_{w^*} (\hat{\mathbf{w}} - \mathbf{w}^*). \end{aligned} \tag{20}$$

The expected value and variance of the difference will be

$$\text{mean}\left[t_i - \hat{a}_i\right] = \text{mean}[\varepsilon_i] - \nabla_{\mathbf{w}}^{\mathrm{T}} a_i \mid_{\mathbf{w}^*} \times \text{mean}\,(\hat{\mathbf{w}} - \mathbf{w}^*) \approx 0$$

$$\text{var}\left[t_i - \hat{a}_i\right] = \text{var}[\varepsilon_i] - \text{var}[\, \nabla_{\mathbf{w}}^{\mathrm{T}} a_i \mid_{\mathbf{w}^*} (\hat{\mathbf{w}} - \mathbf{w}^*)\,] \tag{21}$$

Note that the assumptions that a_i is continuously differentiable and that the matrix $\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{J}(\mathbf{w})$ (with $\mathbf{J}(\mathbf{w}) = \left[\nabla_{\mathbf{w}} a_1 \quad \nabla_{\mathbf{w}} a_2 \quad \cdots \quad \nabla_{\mathbf{w}} a_Q\right]^{\mathrm{T}}$) is nonsingular are essential in the statistical evaluation. The distribution of $\hat{\mathbf{w}} - \mathbf{w}^*$ can be approximated with the distribution, $N_Q\,(0, \sigma^2[\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{J}(\mathbf{w})]^{-1})$, where $\mathbf{J}(\mathbf{w})$ is also the Jacobian matrix (16). In fact, given a large value of Q, ε_i being the random numbers with independent and identical distribution and the parameter space of the weights, $\mathbf{\Omega}$, being a compact subset of n dimensional real number space, the values of the weight vector, $\hat{\mathbf{w}}$, are certain to be within a small neighborhood of the true value vector, $\mathbf{w}^*$. Therefore, in late calculation, the weight vector, $\hat{\mathbf{w}}$, is used to replace $\mathbf{w}^*$ and is written as $\mathbf{w}$.

With the number of samples, Q, and the number of estimated parameters, n, the unbiased estimator of σ^2 is

$$s^2 = \frac{\sum_{i=1}^{Q} (t_i - a_i(\mathbf{x}_i, \mathbf{w}))^2}{Q - n}. \tag{22}$$

Hence, the confidence interval $100\times(1-\alpha)$ for the estimated value, $\hat{a}_i$, will be

$$\hat{a}_i \pm t_{\alpha/2,Q-n}\times s\times(1+\nabla_{\mathbf{w}}^{\mathrm{T}}a_i\{\mathbf{J}^{\mathbf{T}}(\mathbf{w})\mathbf{J}(\mathbf{w})\}^{-1}\nabla_{\mathbf{w}}a_i)^{1/2} \tag{23}$$

where the parameter, α, denotes the level of significance and $t_{\alpha/2,Q-n}$ is the $(1-\alpha/2)$ quartile of a t - distribution with Q-n degrees of freedom.

In order to use the above equation to estimate the likely range of a system output, the model errors should be independently and normally distributed with zero means (Chryssolouris et al., 1996) which can be generally satisfied by practical cases. However, it is also indicated that the confidence bound estimation method is asymptotically valid when a large set of training data is available (Hwang & Ding, 1997). With a small set of training data and relatively large set of parameters, the matrix $\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{J}(\mathbf{w})$ can be singular. In this case, the estimated confidence intervals are unreliable. According to Yang et. al (Yang et al., 2002), the performance index can be changed into the following,

$$E(\mathbf{w})=\sum_{i=1}^{Q}(t_i-a_i(\mathbf{x}_i,\mathbf{w}))^2+\lambda\sum_{i=1}^{n}w_i^2\ , \tag{24}$$

where $\lambda>0$ is a decay parameter. The confidence interval for feedforward neural networks trained by weight decay is

$$\hat{a}_i \pm t_{\alpha/2,Q-n}\times s\times(1+\nabla_{\mathbf{w}}^{\mathrm{T}}a_i\{\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{J}(\mathbf{w})+\lambda\mathbf{I}\}^{-1}\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{J}(\mathbf{w})\{\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{J}(\mathbf{w})+\lambda\mathbf{I}\}^{-1}\nabla_{\mathbf{w}}a_i)^{1/2} \tag{25}$$

3.2 An example of neural network regression

In order to illustrate how a neural network regression model works, an example is taken below. Consider the following function,

$$f(x)=\ 0.5+0.4\sin(2\pi x)+\varepsilon\ , \tag{26}$$

where ε is the random noise normally distributed with the mean of zero and the standard deviation of 0.05. A data set of 21 points with equal intervals between 0 and 1 is chosen as inputs to the function.

For convenience of discussion, the feedforward neural network is chosen as one hidden layer and linear nodes in the output layer. Hence, it can be mathematically represented by the following equation,

$$a_i(x_i)=\mathbf{w}^2\mathbf{h}^1(\mathbf{o}^1)+b^2\ ;\quad \mathbf{o}^1=\mathbf{w}^1x_i+\mathbf{b}^1\ ;\quad i=1,...,21 \tag{27}$$

where x_i and a_i, as defined previously, denote input and output, respectively; $\mathbf{w}^k$ and $\mathbf{b}^k$ $_{(k=1,2)}$, as the same as in Equation (2), are the weights and biases of the network with $\mathbf{w}^2\in\Re^{1\times s_1}$,$\mathbf{w}^1\in\Re^{s_1\times 1}$, $\mathbf{b}^2\in\Re$,$\mathbf{b}^1\in\Re^{\mathbf{s}_1}$ where s_1 is the number of hidden nodes; $\boldsymbol{h}^1(.)$ is the activation function vector in the hidden layer. The activation function $\mathbf{h}^1(\mathbf{o}^1)$ is chosen as a hyperbolic tangent sigmoid function and can be alternatively written by $\mathbf{h}^1(\mathbf{o}^1)=\left[h_1^1(o_1)\quad h_2^1(o_2)\quad\cdots\quad h_{s^2}^1(o_{s^2})\right]^{\mathrm{T}}$ with $h_i^1(o_i)=\tanh(o_i)=(e^{o_i}-e^{-o_i})/(e^{o_i}+e^{-o_i})$.

In order to train the neural network, the initial weights and biases are random numbers uniformly distributed between -1 and 1. The Levenberg-Marquardt backpropagation

algorithm is used to train the neural network. The initial value of μ is chosen as 0.001. The parameter γ is chosen as 10. The number of epochs is 1000. Equation (25) is used to calculate the prediction interval, where the parameter, λ, is chosen as 0.0001. And 95% confidence level is used for the simulation.

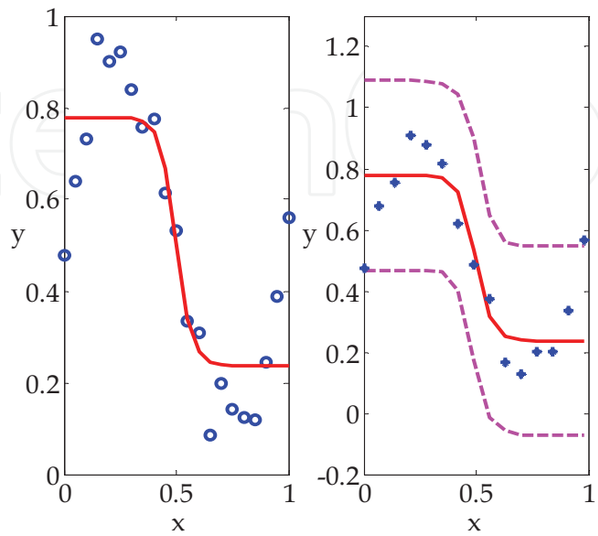


Fig. 2. Neural model of $f(x)$ with one hidden node. Left figure: circles - training data; solid line - neural network output. Right figure: asterisks - testing data; solid line - neural network output; dashed lines - 95% confidence interval

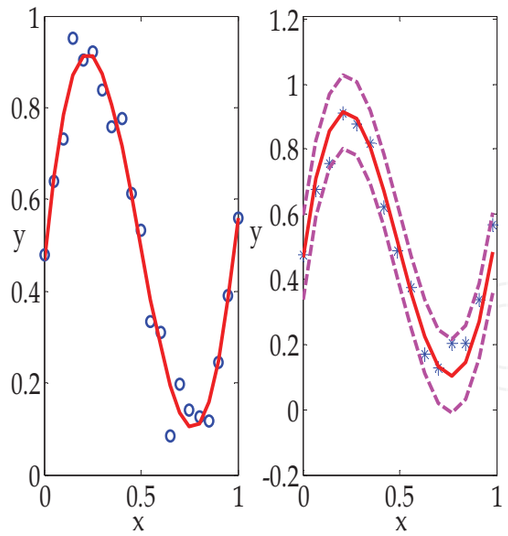


Fig. 3. Neural model of $f(x)$ with two hidden nodes. Left figure: circles - training data; solid line - neural network output. Right figure: asterisks - testing data; solid line - neural network output; dashed lines - 95% confidence interval

Fig. 2 shows the training data points and the neural network output with one hidden node (left figure). After training, the neural network model is used to predict the output of 15 unseen inputs which are corrupted with normally distributed noise of $N(0, 0.05^2)$. The right

side of Fig. 2 provides the predicted output of the neural network, which is drawn in solid line, the testing data (in asterisk symbol) and the confidence interval (in dashed line) with 95% confidence level. As can be seen, the neural network model provides a wide prediction range as a consequence of limited capability of the neural network with only one hidden node.

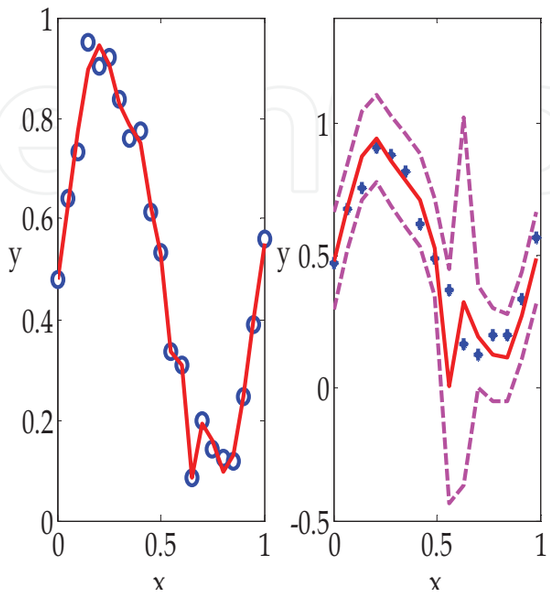


Fig. 4. Neural model of $f(x)$ with five hidden nodes. Left figure: circles - training data; solid line - neural network output. Right figure: asterisks - testing data; solid line - neural network output; dashed lines - 95% confidence interval.

The simulation was repeated 100 times. After each simulation, the maximal predicted interval of 15 points was recorded. The average of the maximal predicted intervals for 100 times of simulations is 0.7524. By comparing to the range of the function $f(x)$, which is between 0 and 1, the predicted interval is found to be too wide. A better fit and prediction ability of the neural network can be obtained by increasing the number of hidden nodes.

Fig. 3 shows the neural network output with two hidden nodes, which gives a much better approximation to $f(x)$. The average of maximal predicted intervals over 100 times of simulations is 0.2889. However, if the number of hidden nodes is too large, then, the error due to approximation to the underlying function becomes worse. Fig. 4 shows the result of fitting the function $f(x)$ using the neural network that contains five hidden nodes. Since the neural network has fitted the data by developing some dramatic oscillations, it eventually provides a poor prediction of $f(x)$ with wide confidence interval at some points, where the neural network fitting to noisy data points can be seen.

In order to examine how the number of hidden nodes impacts the prediction interval, the number of hidden nodes was chosen to be one, two, three, four and five. For each case, the simulation was repeated 100 times, and the average of the maximal prediction intervals were calculated accordingly. The results are shown in Table 1. From Table 1, it can be observed, that, for this example, the neural network with two hidden nodes provides the best prediction. When the number of hidden nodes increases, redundant nodes exist in the neural network, which lead the neural network to overfit some noisy data. Consequently, a wide confidence interval at some points indicates imprecise prediction.

Number of hidden nodes	1	2	3	4	5
Average of maximal prediction intervals by 100 times simulations	0.7524	0.2889	0.3117	0.3149	5.3223

Table 1. Average of maximal prediction intervals versus the number of hidden nodes

4. Modeling and prediction of nonlinear elastic behavior of reinforced soil

In this section, a practical application of feedforward neural networks to modeling of nonlinear behavior of composite materials will be presented. In particular, soil reinforced with geofiber and lime powder is taken as a composite material to be investigated as an example of such an application.

Mechanical behavior of soil under static and dynamic loading plays an important role in performance of infrastructures such as durability of pavement and road beds, and stability of slopes and bridge foundations, etc. To improve engineering properties, soil reinforced with other materials becomes widely applied in geotechnical engineering (Michalowiski & Zhao, 1995, and Li & Ding, 2002). Although the performance can be significantly improved when soil is reinforced with short fiber and stabilized with lime powder, quantitative evaluation of enhancement of soil mechanical behavior is still difficult since the stress-strain relation is highly nonlinear and sensitive to various factors such as lime and fiber contents, confining pressures, sample curing periods mixed with lime, etc. (Li & Zhang, 2003).

Traditionally, modeling of engineering materials was conducted by taking the following three steps. First, a constitutive model (e.g., a nonlinear elastic model) needs to be established. Second, constitutive parameters are identified and calibrated with experimental data using a conventional method (e.g. linear regression). Third, the constitutive model needs to be validated using experimental data from a laboratory or field (e.g., shearing tests). Since the constitutive parameters are nonlinear functions of multiple variables, a traditional approach cannot calibrate the parameters accurately and efficiently. In particular, when soil is mixed with lime powder and fibers, the constitutive parameters become a function of many interrelated variables. Under the circumstance, the coupling effects among different variables may significantly impact on the relationship of stress and strain. The coupling effects, however, cannot be practically described in a traditional model due to their intrigued nature and significant amount of experimental work. In this section, it is proposed that the nonlinear elastic behavior of composite soils is to be modeled using a feedforward neural network.

Applying neural network regression to modeling of reinforced soil is a new research topic. Till now, very few discussions of the potential application of neural networks in civil engineering have been found, for instance, modeling of shear strength of reinforced concrete beams (Rajasekaran & Amalraj, 2002) and estimation of resilient modulus of aggregate base using a feedforward neural network (Issa & Zamam, 1999). Therefore, relatively detailed description of the necessary knowledge regarding experimental investigation and data acquisition is provided in this section. Then, neural network training as well as the prediction using a neural network regression model with unseen inputs will be presented, which provides statistical justification for the case analysis and decision making in construction using the neural network model. Finally, the parameter sensitivities to the inputs such as fiber and lime contents, confining pressure, sample aging period are analyzed based on the neural network regression model (He & Li, 2008 and 2006).

4.1 Problem background and experiment setup

4.1.1 Shear stress-strain relation

The nonlinear elastic behavior exhibited by soil mixed with short fiber and lime powder can be affected by many factors. The shear stress-strain relation of soil skeleton in terms of the second invariants of deviatoric tensors in a three-dimensional space can be expressed by

$$\sigma^s = E\varepsilon^s \tag{28}$$

where ε^s denotes a shear strain invariant related to the second invariant of a deviatoric stress tensor; σ^s denotes a shear strain invariant associated with the second invariant of a deviatoric strain tensor. E is the shear modulus and a function of mechanical properties of the reinforced soil such as the initial shear modulus and strength of reinforced soil.

Since the objective of experiment under the investigation aims to understand the mechanical behavior of subgrade soil reinforced with short fiber and stabilized with lime, consequently, the shear modulus, E, is assumed to be a function of multi-variables, such as shear strain, confining stress, fiber and lime contents. Moreover, when the soil samples are mixed with lime powder, the curing time (or aging period) before a shearing test will be an auxiliary variable with lime content.

For convenience of experimental investigations using a conventional triaxial apparatus, it is necessary to simplify the stress-strain relation (28) from a true three-dimensional space to an expression in a quasi three-dimensional space. In the simplified space, the stress-strain relation in Equation (28) reduces to

$$\sigma_a = E\ (\sigma_0, \varepsilon_a, \beta_F, \beta_L, t)\ \varepsilon_a \tag{29}$$

where σ_0 is confining pressure; σ_a and ε_a denote the principal stresses and strains in three-dimension that are individually simplified from the invariants of the deviatoric stress and strain tensors; β_F and β_L are contents of fiber and lime, respectively; t is the curing time of soil sample before shear testing. For conventional triaxial shearing tests, σ_a and ε_a can simply represent the axial stress and strain respectively. To provide the input data for training and validating a feedforward neural network model, experimental tests need to be conducted in laboratory first so that the stress-strain relationship in Equation (29) can be determined.

4.1.2 Experiment and testing data

In laboratory, nine groups of unsaturated and reinforced soil samples were subjected to triaxial shearing tests. The tested soil has the following physical properties: the wet unit weight γ_{wet} = 16.66 kN/m^3; plastic limit PL = 5%; the soil classification (AASHTO) is A4. The specimen is cylindrical with a dimension of 6.86 cm in diameter and 13.72 cm in height. The sample preparation and testing followed the AASHTO code T297 (or ASTM D4767) with special consideration for the procedure of mixing short geofiber (5 cm) with soil. Nine groups of soil specimens were prepared with β_F = 0%, 0.2% and 0.5%; β_L = 0% and 5%; and t =1, 7, 14 and 28 days before shearing tests. Unsaturated specimens were tested under a consolidated-undrained condition using a conventional triaxial apparatus. The controlled shear loading rate was 0.006 min^{-1}. Four different confining pressures (σ_0 = 50, 100, 150, and 200 kPa) were applied to specimens in each group. The combination of selected fiber contents, lime contents, confining pressures and aging periods of soil specimens generates thirty four sets of experimental setup which is listed in Table 2.

Testing results from 34 shear tests were collected and processed through a data acquisition system. For purpose of demonstration, testing curves of 30 stress-strain (σ_a -ε_a) relations are

Set No.	Test No.	β_F(%)	β_L(%)	σ_0 (kPa)	t(day)
1	S11	0	0	50.0	1
2*	S12	0	0	100.0	1
3	S13	0	0	150.0	1.0
4	S14	0	0	200.0	1.0
5	F5	0.2	0	50.0	1.0
6	F6	0.2	0	100.0	1.0
7	F7	0.2	0	150.0	1.0
8	F8	0.2	0	200.0	1.0
9	F5-5	0.5	0	50.0	1.0
10	F5-6	0.5	0	100.0	1.0
11	F5-7	0.5	0	150.0	1.0
12	F5-8	0.5	0	200.0	1.0
13	Slf2	0.2	5.0	50.0	7.0
14	Slf1,11, 9	0.2	5.0	100.0	7.0
15	Slf6,17	0.2	5.0	150.0	7.0
16	Slf3,15	0.2	5.0	200.0	7.0
17	Slf5,12,16	0.2	5.0	50.0	14.0
18	Slf4	0.2	5.0	100.0	14.0
19	Slf7,8	0.2	5.0	150.0	14.0
20	Slf14	0.2	5.0	50.0	28.0
21*	Slf13	0.2	5.0	100.0	28.0
22	Slf18,19,20	0.2	5.0	150.0	28.0
23*	Slf5-1	0.5	5.0	50.0	7.0
24	Slf5-2	0.5	5.0	100.0	7.0
25	Slf5-4,5-10	0.5	5.0	150.0	7.0
26	Slf5-6,5-12	0.5	5.0	200.0	7.0
27	Slf5-3	0.5	5.0	50.0	14.0
28	Slf5-5	0.5	5.0	100.0	14.0
29	Slf5-7, 5-11	0.5	5.0	150.0	14.0
30	Slf5-8	0.5	5.0	200.0	14.0
31	Slf5-16	0.5	5.0	50.0	28.0
32	Slf5-13	0.5	5.0	100.0	28.0
33	Slf5-14	0.5	5.0	150.0	28.0
34*	Slf5-15	0.5	5.0	200.0	28.0

Table 2. Experimental Conditions

drawn in Fig. 5. In the figure, solid squares, down-triangles, asterisks and five-point stars represent experimental data with confining pressures, σ_0, of 50, 100, 150 and 200 kPa, individually. In each subfigure of Fig. 5, the chosen fiber content β_F, lime content β_L, as well as the curing period of soil sample t, are provided at the bottom of each subfigure. From the subfigures, it can be seen that the strength of reinforced soil can be improved by increasing the fiber content, β_F, without adding lime powder (β_L = 0%) and holding the aging period of 1 day. If the fiber content changes from 0% to 0.5%, the maximum value of axial stress, σ_a, can vary from 280 kPa to 500 kPa (see subfigures 1~3 of Fig. 5). Furthermore, by adding lime and prolonging the aging period, the axial stress (soils strength) can be notably enhanced

(see subfigures 4~6 of Fig. 5). Similarly, soil elastic modulus and strength are also evidently improved with increasing the fiber content β_F (see subfigures 7~9 of Fig. 5).
Testing results in Fig. 5 indicate high nonlinearity between the axial stress and axial strain affected by four variables. To describe such mechanical behavior, a feedforward neural network model is used to predict the nonlinear relationship between multiple inputs and the output.

4.2 Modeling and predicting nonlinear elastic behavior of reinforced soil

As indicated in Equation (28), the axial stress, σ_a, is a nonlinear function of variables σ_0, β_F, β_L, t and the axial strain, ε_a. The function is to be approximated using a feedforward neural netowork. To validate the results of neural network regression, the prediction confidence of soil deformation using the feedforward neural network is analyzed.

4.2.1 Modeling of the reinforced soil

To train the feedforward neural network, the variables of fiber content β_F, lime content β_L, confining pressure σ_0 , sample curing period t and axial strain ε_a are applied as inputs to the

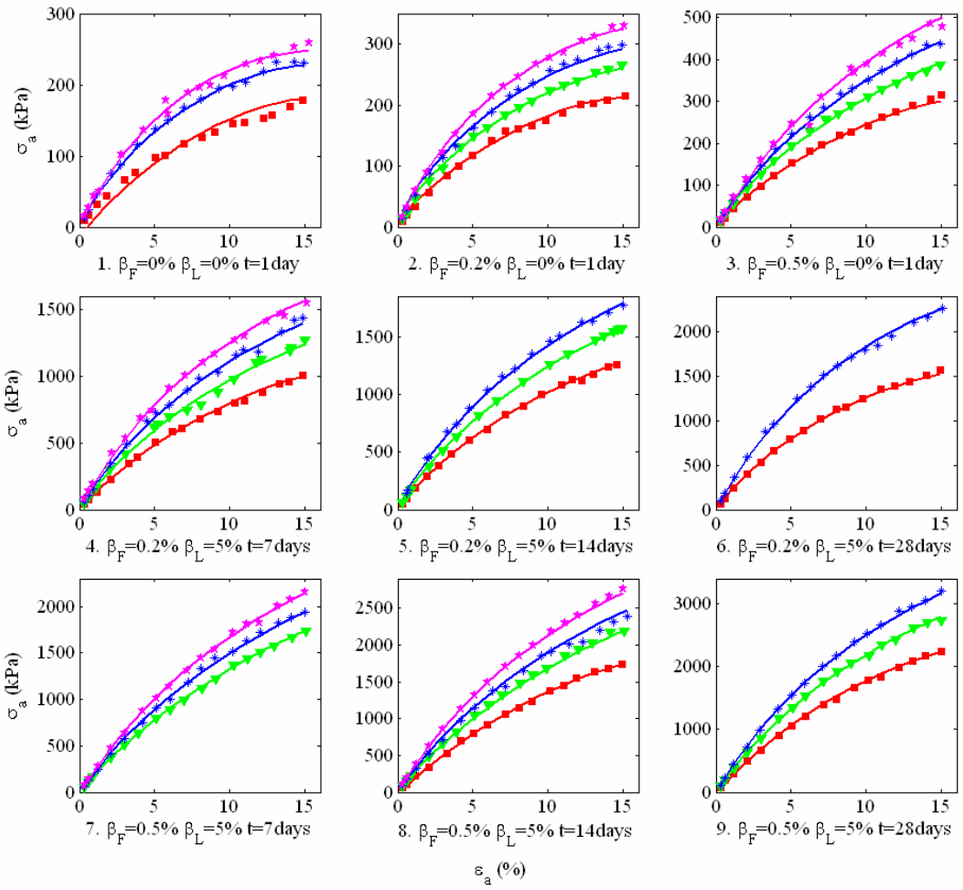


Fig. 5. Relationship of σ_a vs. ε_a with different confining pressures (CP) (Experiment data: Symbols with CP values: ■50 kPa, ▼100 kPa, ✷150 kPa,★200 kPa; The outputs of the neural network are in solid lines)

neural network while the corresponding axial shear stress, σ_a, is designated as the output. Thirty four sets of data in Table 2 (Seventeen axial strain values in each set within a range from 0.3% to 15.25%) generate 578 (=34×17) input patterns. Accordingly, 578 measured values of the axial shear stress are the targets for neural network to learn and test. As presented in Fig. 5, thirty sets of data which produces 510 input/output patterns (30×17=510) are used to train the neural network, whereas the remaining 68 input/output patterns (4 sets) marked with the asterisk sign, *, in Table 2 are applied to test the neural network model. The testing data are chosen to represent low, median and high nonlinearity of soil mechanical behavior.

Without loss of generality, one-hidden-layer feedforward neural network with linear output layer is chosen. In addition, in order to make the neural network converge quickly, the scaling factors 0.01, 10, 1.0, 0.1 and 0.1 are used for the inputs σ_0 (kPa), β_F (%), β_L (%), t (day) and ε_a (kPa), respectively. Similarly, the axial shear stress, σ_a, is scaled by a factor of 0.0001. Thus, the input vector for training the neural network is $x = [0.01\sigma_0, 10\beta_F, \beta_L, 0.1t, 0.1\varepsilon_a]^T$ and the desired output is $0.0001\sigma_a$. With the input, $\mathbf{x}$ and the output, $N\sigma_a$, the neural network model is defined as follows,

$$N\sigma_a(\mathbf{x}_i) = \mathbf{w}^2\mathbf{h}^1(\mathbf{o}^1) + b^2 \; ; \quad \mathbf{o}^1 = \mathbf{w}^1\mathbf{x}_i + \mathbf{b}^1 \; ; \quad i=1,...,510 \tag{30}$$

As discussed in Section 3, the number of hidden nodes should be chosen so that the neural network model will not overfit the noisy data. After several times of pretraining and prediction capability analysis, the number of hidden nodes has been accordingly chosen as 6.

The Levenberg Marquardt backpropagation training algorithm is chosen to train the neural network. The parameters are chosen as follows: the weights and biases are initialized with the random numbers uniformly distributed between -1 and 1; the maximal number of training epochs is 1,500; the error goal is that the sum of squared errors (SSE) is less than or equal to 0.002. The adaptive factor, μ, is initialized as 0.0001 and the update factor, γ, is assumed to be 10. After 702 epochs, the sum of squared errors (SSE) monotonically decreases to 0.00199. After the training, the outputs from the neural network are plotted in Fig. 5 in solid lines along with testing data.

4.2.2 Approximation error and prediction performance

As aforementioned, the most important criterion to evaluate a neural network model is its capacities of generalization and prediction. The prediction intervals for the trained neural network (30) are calculated and analyzed.

Before Equation (30) is used to predict the axial stress, σ_a, with unseen values of the variables σ_0, β_F, β_L, t and ε_a, the modeling errors between the experimental data and the neural network regression model are computed. The errors are sorted with the interval of 20 (i.e., -120, -100, ...,-20, 0, 20, …,100, 120) and depicted in Fig. 6. The mean value of the errors is 0.05 kPa. Compared to the shear stress values of the experiment, which are between 11.2 kPa and 3667.8 kPa, the mean value is small enough to be considered as zero. The standard deviation of the normal distribution is 19.8 kPa. The unbiased estimator, s, is 20.69 kPa.

With the trained weights, when the testing input patterns are fed into the neural network, the corresponding outputs are predicted. The mean value and standard deviation of errors between experimental data and the outputs of the neural network are 12.2 kPa and 39.5 kPa, respectively, which is relatively larger than the errors provided by training data.

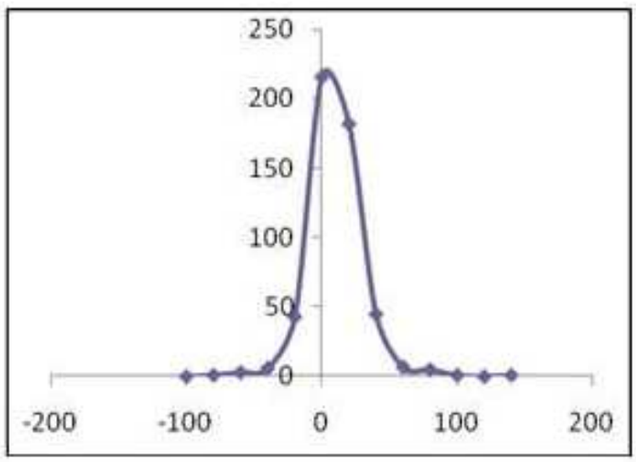


Fig. 6. Distribution of the errors between desired outputs (from 510 experimental data) and corresponding neural network outputs

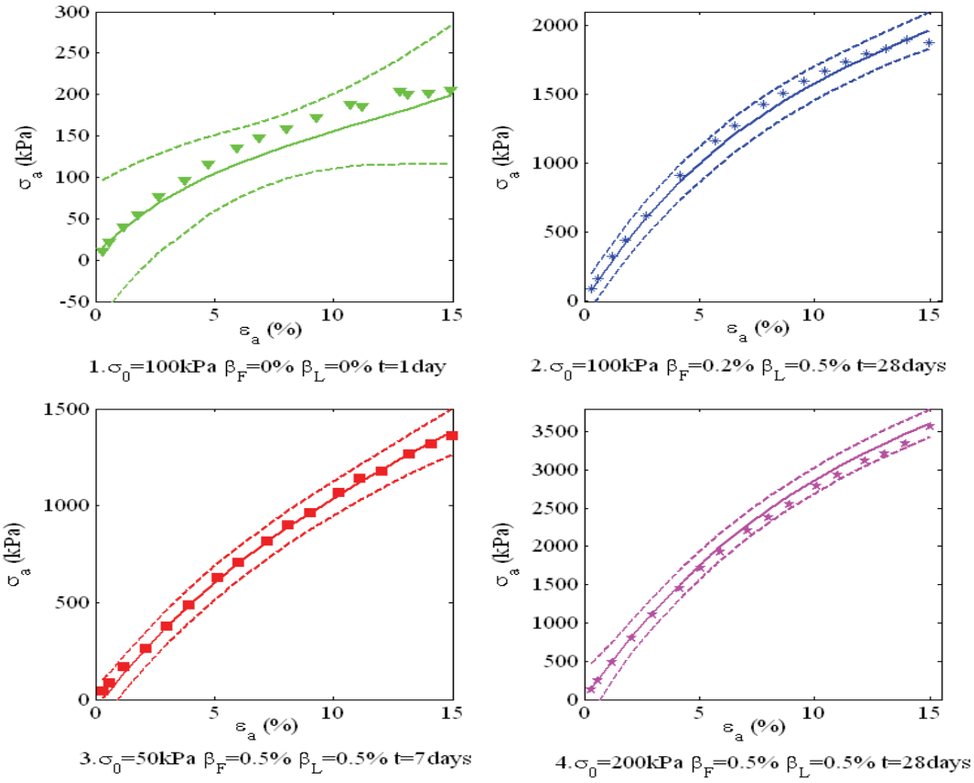


Fig. 7. Testing data with 95% confidence intervals (Experimental data are drawn with the inverse triangle, asterisks, solid squares and five-point stars; the outputs of the neural network are drawn in solid lines. The confidence levels are drawn in dashed lines.)

According to Equation (23), the confidence intervals (with 95% confidence level) are calculated. Dashed lines in Fig. 7 show the envelope of confidence intervals of predicted outputs along the increase of axial shear strain for four sets of data. Actual experimental data are filled in the figure with down-triangles, asterisks and five-point stars and solid

squares, respectively. Although some points are close to certain lower bound values, e.g., the outputs of the neural network around 3,000 kPa in subfigure 4 of Fig. 7, all testing data are within its upper and lower bound values. The confidence intervals provide the envelope for prediction of a shear stress output.

4.2.3 Sensitivity analysis of neural network-based parameters

In opposition to the conventional model-based nonlinear technique, a neural network generalizes a model by learning from experimental data, which is particularly important when the underlying relationships of a researched object are unknown. Using the trained neural network model, soil mechanical behavior can be quantitatively analyzed with limited experimental data.

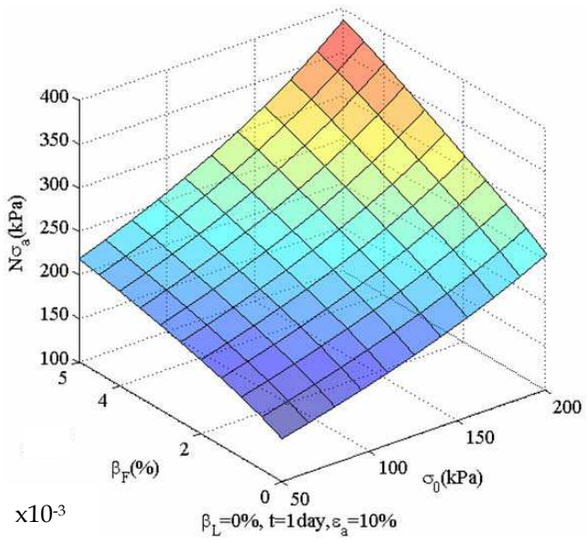


Fig. 8. Axial stress vs. confining pressure and fiber content

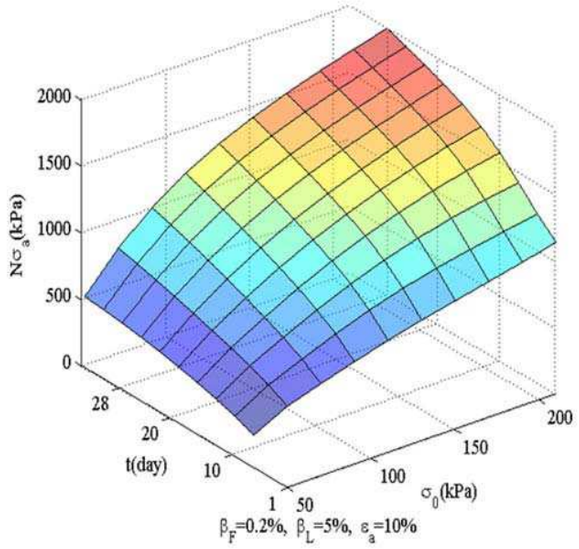


Fig. 9. Axial stress vs. confining pressure and sample aging period

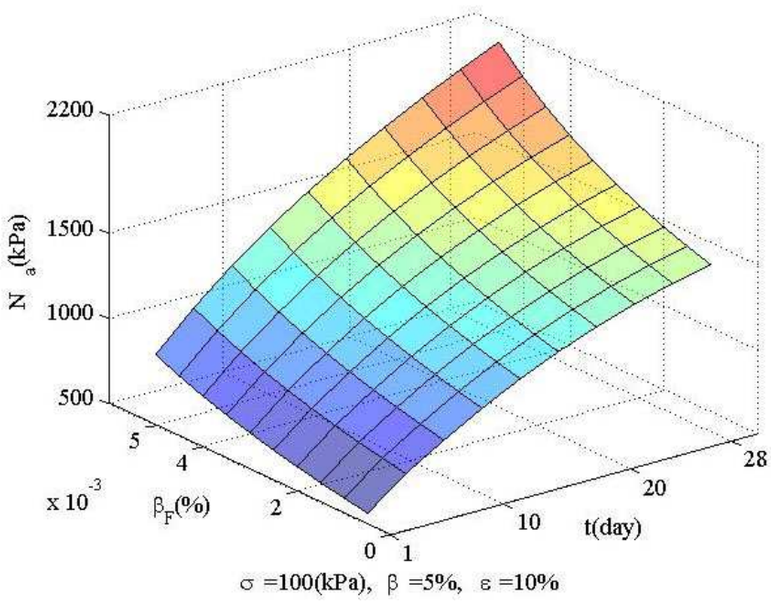


Fig. 10. Axial stress vs. fiber content and sample aging period

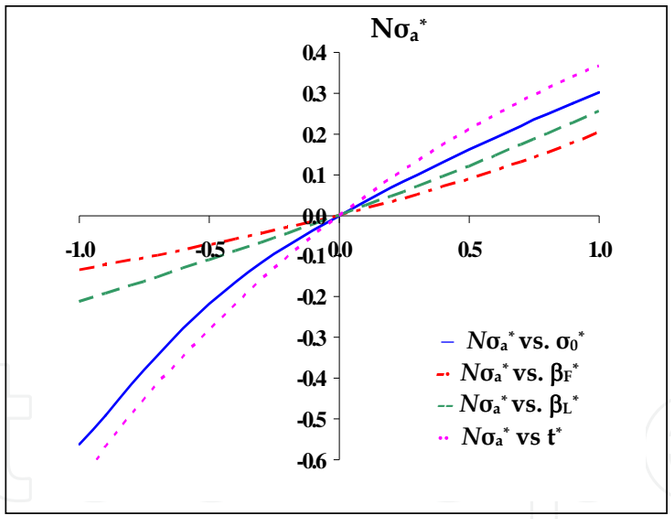


Fig. 11. Sensitivity analysis with dimensionless relations between the axial shear stress and variables (σ_0^*, β_F^*, β_L^*, t^*)

1. Soil mechanical behavior in response to variable coupling effects

The diagrams in Figs. 5 and 7 provide a set of σ_a ~ε_a relations. Beyond the relations, one may be interested in how the variables like fiber content and lime content as well as aging period co-influence the strength of reinforced soil. In order to investigate parameter coupling effects, the axial strain at 10% is assumed for convenience of analysis. Based on the assumption, joined impacts of multiple variables on axial shear stress are calculated and illustrated by Figs. 8, 9 and 10, respectively.

Firstly, in Fig. 8, a joined impact of confining pressure, σ_0, and fiber content, β_F, on the estimated axial shear stress (i.e. $N\sigma_a$ in Fig. 8) is presented for given lime content (β_L= 0%) and sample curing period t (1 day). According to Fig. 8, if no fiber is added to the soil sample (i.e. β_F = 0%), the axial stress appears linearly increased when the confining pressure changes from 50 kPa to 200 kPa. However, it can be observed that when 0.5% short fiber is mixed with the soil, the axial stress increases exponentially with the change of the confining pressure. This suggests that the stress-strain relation is more sensitively in response to the fiber content than to the confining pressure though both of the two variables have significant impact in soil mechanical behavior.

Secondly, the joined impact of confining pressure, σ_0, and, aging period, t, on soil stress, σ_a, due to adding lime (β_L=5%) is examined. Fig. 9 presents the increase of estimated stress value verse curing time of the lime mixed soil and confining pressure (with fiber content of 0.2%). Again it can be noted that the estimated stress curve, $N\sigma_a$, is nonlinearly changed with the variations of aging period and confining pressure. When aging period is short, e.g. 1 day, adding lime powder seems to have less influence on axial shear stress. The value of axial shear stress will be increased about 290 kPa (from 480 kPa to 770 kPa) with confining pressure raising from 50 kPa to 200 kPa. However, when aging period is extended to 28 days, the variation of the axial stress along with the same range of confining pressure raises to 960 kPa (from 1,130 kPa to 1,990 kPa). This indicates that soil strength can be largely improved by prolonging aging period. In addition, compared to Fig. 8, where the range of axial shear stress is between 100 kPa and 400 kPa, the value of axial shear stress in Fig. 9 can be increased from 500 kPa to 2,000 kPa. This implies that axial resistance between soil granular particles has been substantially enhanced by the extra bonding force due to chemical stabilization because of adding lime powder.

Although Fig. 9 exhibits a significant contribution of lime powder with a longer aging period to the strength of the reinforced soil, it can also be observed that the axial stress is increased logarithmically along the aging period.

Finally, Fig. 10 is used to illustrate the combined impact of the fiber contents and 5% lime content with different aging periods on nonlinear soil stress-strain relations. For given lime content β_L = 5 % and confining pressures σ_0 = 100 kPa, the change of axial shear stress along with different curing periods and fiber contents is shown in the figure. Besides highly nonlinear relation between axial shear stress and aging period and fiber content, it can be clearly observed that the values of axial stress increase exponentially with the increase of fiber content, whereas it increases logarithmically with the increase of aging period. This indicates the short fiber with additional tensile and shear resistance may play a more important role on further improvement of strength of the soil since an exponential function can increase much faster than a logarithmic function with same amount of input.

The results presented in Figs. 8, 9 and 10 indicate: 1) the axial shear stress is a nonlinear function of multiple variables σ_0, β_F, β_L, t and σ_a; 2) the strength of soil can be improved significantly by adding short fiber and lime powder with an aging period; 3) the axial shear stress increases exponentially with the increase of fiber content and logarithmically with a prolonging aging period when mixed with lime powder.

2. Soil mechanical behavior in response to an individual variable

In above section, the coupling effect among confining pressure, fiber content and lime content as well as aging period are qualitatively analyzed and tendencies of parameter

changes are depicted. Furthermore, sensitivity of axial shear stress to each of the variables can be quantitatively analyzed. To present the work clearly, the neural network based axial stress-strain relation and relative variables are defined as follows,

$$N\sigma_{\mathrm{a}} = N\sigma_{\mathrm{a}}(\sigma_0, \beta_{\mathrm{F}}, \beta_{\mathrm{L}}, t, \varepsilon_{\mathrm{a}}) . \tag{31}$$

where $N\sigma_{\mathrm{a}}$ represents the axial shear stress predicted by the feedforward neural network to distinguish the notations of the axial stress σ_{a} in early sections. For purpose of investigation, the axial strain at failure is assumed (ε_{a} =10%) and the initial values of other variables in (31) are individually chosen as σ_{0i} =100 kPa, $\beta_{\mathrm{F}i}$=0.25%, $\beta_{\mathrm{L}i}$ =2.5% and t_i =14 days where the subscript i denotes the initial value. Also for convenience of sensitivity analysis, the normalized deviations from initial input values and the corresponding outputs defined as the following form: $X^* = (X - X_i)/X_i = \Delta X/X_i$ and $Y^* = (Y - Y_{\mathrm{i}})/Y_{\mathrm{i}} = Y/Y_{\mathrm{i}} - 1$ where X and Y represent the inputs (i.e., σ_0, β_{F}, β_{L} or t) and the output [i.e., $N\sigma_{\mathrm{a}}(\sigma_0, \beta_{\mathrm{F}}, \beta_{\mathrm{L}}, t)$]; X_i and Y_i stand for the initial values of X and Y. The normalized deviation of input variables can be alternatively written by $X^* = \Delta X/X_i$ that changes from -1 to +1 when the incremental value ΔX (i.e. $\Delta\sigma_0$, $\Delta\beta_{\mathrm{F}}$, $\Delta\beta_{\mathrm{L}}$ or Δt) varies from $-X_i$ to $+X_i$ as the range chosen for sensitivity analysis. The dimensionless relation below exemplifies how the axial shear stress responds to the confining pressure:

$$N\sigma_{\mathrm{a}}{}^*(\sigma_0^*) = \frac{N\sigma_{\mathrm{a}}(\sigma_{0i}(1+\sigma_0^*), \beta_{\mathrm{F}i}, \beta_{\mathrm{L}i}, t_i, \varepsilon_{\mathrm{a}})}{N\sigma_{\mathrm{a}}(\sigma_{0i}, \beta_{\mathrm{F}i}, \beta_{\mathrm{L}i}, t_i, \varepsilon_{\mathrm{a}})} - 1 \tag{31}$$

$$\sigma_0^*(\sigma_0) = (\sigma_0 - \sigma_{0i}) / \sigma_{0i} = \Delta\sigma_0 / \sigma_{0i} \tag{32}$$

The relation $N\sigma_{\mathrm{a}}{}^*(\sigma_0{}^*)$ in (31) against $\sigma_0{}^*$ in (32) is graphically drawn in a solid blue line in Fig. 11. Similar to (31) and (32), the dimensionless relation Y^* vs. X^* for other variables such as $N\sigma_{\mathrm{a}}{}^*(\beta_{\mathrm{F}}{}^*) \sim \beta_{\mathrm{F}}{}^*$, $N\sigma_{\mathrm{a}}{}^*(\beta_{\mathrm{L}}{}^*) \sim \beta_{\mathrm{L}}{}^*$, and $N\sigma_{\mathrm{a}}{}^*(t^*) \sim t^*$ can be found accordingly and plotted in the same figure.

The results in Fig. 11 display how sensitive the axial shear stress is when changing confining pressure, fiber content, lime content or aging period. In comparison of the relations $N\sigma_{\mathrm{a}}{}^* \sim \beta_{\mathrm{L}}{}^*$ (in a dash line) and $N\sigma_{\mathrm{a}}{}^* \sim t^*$ (in a dot line) with $N\sigma_{\mathrm{a}}{}^* \sim \beta_{\mathrm{F}}{}^*$(in a dash-dot line), it can be observed that the soil axial shear stress is more responsive to the lime content than to the fiber content, and most sensitive to the soil-lime curing period among the four input variables. For instance, for the given axial strain and initial values (σ_{0i} = 100 kPa, $\beta_{\mathrm{F}i}$ = 0.25%, $\beta_{\mathrm{L}i}$ = 2.5% and σ_{a} = 10%), if $\beta_{\mathrm{F}}{}^*$, $\beta_{\mathrm{L}}{}^*$, and t^* increase by 100% (X^* = 1.0) from their initial values, the soil axial strength is improved by 20%, 25% and 35%, respectively. The quantitative results in Fig. 11 are consistent with the coupling effects shown in Figs. 8, 9 and 10. The fact that confining pressure substantially affects soil strength can be also observed in Fig. 11.

5. Summaries and conclusions

In this chapter, the standard backpropagation algorithm and the Levenberg-Marquardt backpropagation algorithm were derived in vector forms. Then, the confidence intervals

and prediction intervals for the nonlinear structure like feedforward neural networks were discussed. Particularly, the impact of neural network structure, i.e. the number of hidden nodes, to confidence intervals was analyzed and demonstrated via a simple example. Finally, modeling of nonlinear elastic behavior of the reinforced soil using a feedforward neural network was conducted. As an application, the sensitivity of the strength of reinforced soil to the constitutive parameters was analyzed using the neural network-based model. From the work presented in this chapter, the following conclusions can be drawn:

1. The standard backpropagation algorithm and the Levenberg-Marquardt backpropagation algorithm were derived in vector forms. The vector forms of the backpropagation algorithms make training neural networks and computing confidence intervals more efficient and less error prone.
2. Confidence intervals and prediction intervals for neural network regressions were presented. Especially, when the Levenberg-Marquardt backpropagation algorithm is used to train a neural network, since the Jacobian matrix has been calculated to update the weights and biases of the neural network, the confidence interval with a confidence level can be easily computed to evaluate the predictive capability of the neural network which the unseen data is fed into. A demonstrated example shows that too many hidden nodes in a neural network may result in a poor prediction, i.e. the prediction interval will be too wide since the trained neural network overfits noisy data. Meanwhile, not enough hidden nodes will also lead a poor prediction since the nonlinearity of a model cannot be fully identified.
3. Modeling of nonlinear elastic behavior of the soil reinforced with short fiber and stabilized with lime powder using a feedforward neural network was performed. This is the first attempt to model the nonlinear elastic behavior of fiber-lime reinforced soil under multi-axial shear loading using a neural network. The results of modeling reinforced soil are satisfactory. From the experimental data, neural network model and prediction intervals calculation, the following three points can be summarized and concluded,

a. Testing results indicate that the axial stress-strain relation is a nonlinear function of multiple variables, such as confining pressure, fiber content, lime content and sample curing time. To simulate such a nonlinear stress-strain relationship, a feedforward neural network is a good tool. The adopted neural network has one hidden layer with six nodes in it. Five variables are designated as inputs to model nonlinear elastic behavior of the soil. To train and test the neural network, thirty sets of data from conventional triaxial shear tests are selected to train the neural network and four sets of unseen data are adopted to evaluate the trained neural network. Using the derived approximate confidence interval equation (23), all predicted values of axial shear stress by the neural network model are within the envelope of the confidence interval with confidence level of 95%.
b. Parameters sensitivity and coupling effect were analyzed using the neural network based model. The sensitivity analysis of soil mechanical property to the model inputs such as the fiber content, the lime content, confining pressure and the sample curing period was conducted. The quantitative results show that the soil mechanical property is more sensitive to the lime content than to the fiber content. The mechanical property of the soil-lime mixture can be substantially improved with a prolonging curing period,

particularly within a duration from 0 to 28 days. From the analysis of coupling effect, the axial shear stress of composite soil increases exponentially with the increase of fiber content and logarithmically with an increasing aging period for the soil mixed with lime powder. Sample curing time plays a more significant role than other factors.

c. The neural network model provides a convenient and useful tool for analysis of mechanical behavior of composite soil and for applications to various engineering designs. With the confidence interval evaluation, the neural network model can be further applied to the stress-strain relation for large soil deformation

6. Appendix

Neural networks derive their advantages from their special structures – the massive interconnection of simple processing units. If the weights and biases of a neural network are considered as elements of matrices, the matrix calculus will become very useful for developing new algorithms for training neural networks. This appendix provides two basic chain rules for matrix derivatives. Detailed information can be found in the books authored by Cichocki and Unbehauen (Cichocki and Unbehauen, 1993) and Lewis (Lewis, 1995).

1. **The general chain rule**

Theorem 1 Let $\mathbf{f}: D \to \Re^m$ *be a differentiable real vector on an open r-dimensional set* $D \subset \Re^r$, *and let* $\mathbf{u}: S \to D$ *be differentiable on an open set n-dimensional* $S \subset \Re^n$. *Then, the composite vector function* $\mathbf{F}(x)=\mathbf{f}(u(\mathbf{x}))$ *is differentiable on the open set S. The general chain rule of the differentiation of the vector function F(x) is*

$$\partial \mathbf{F} / \partial \mathbf{x} = \mathbf{J}_{\mathbf{F}}(\mathbf{x}) = \mathbf{J}_{\mathbf{f}}(\mathbf{u}(\mathbf{x}))\mathbf{J}_{\mathbf{u}}(\mathbf{x}) \tag{A.1}$$

and the gradient matrix $\nabla_{\mathbf{x}}\mathbf{F}$ *is*

$$\nabla_{\mathbf{x}}\mathbf{F} = \nabla_{\mathbf{x}}\mathbf{u}\ \nabla_{\mathbf{u}}\mathbf{f} \tag{A.2}$$

Proof is omitted.

2. **The chain rule for differentiation of a scalar function with respect to a matrix**

Theorem 2 *Let* $F = h(\mathbf{f})$ *be a differentiable real-valued scalar function of a real vector* $\boldsymbol{f}$, *h:* $\Re^m \to \Re$ *and let* $\mathbf{f}(\mathbf{w}) = [\, f_1(\mathbf{w}) \quad f_2(\mathbf{w}) \quad \cdots \quad f_m(\mathbf{w})\,]^{\mathrm{T}}$ *be a differentiable vector function of the matrix* $\boldsymbol{w}$ *with*

$$\mathbf{w} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1l} \\ w_{21} & w_{22} & \cdots & w_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nl} \end{bmatrix}.$$

Then, the chain rule for the differentiation of the scalar function F with respect to the matrix $\mathbf{w}$ *yields*

$$\frac{\partial F(\mathbf{w})}{\partial \mathbf{w}} = \sum_{k=1}^{m} \frac{\partial h}{\partial f_k(\mathbf{w})} \frac{\partial f_k(\mathbf{w})}{\partial \mathbf{w}} \tag{A.3}$$

Proof: from the derivative of the scale function with respect to a variable matrix, one has

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial f}{\partial w_{11}} & \frac{\partial f}{\partial w_{12}} & \cdots & \frac{\partial f}{\partial w_{1l}} \\ \frac{\partial f}{\partial w_{21}} & \frac{\partial f}{\partial w_{22}} & \cdots & \frac{\partial f}{\partial w_{2l}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial w_{n1}} & \frac{\partial f}{\partial w_{n2}} & \cdots & \frac{\partial f}{\partial w_{nl}} \end{bmatrix} \tag{A.4}$$

For each element of Equation (A.4), use the general chain rule to reach $\nabla_{w_{ij}} F = \nabla_{\mathbf{f}} F \, \nabla_{w_{ij}} \mathbf{f}$, which can be alternatively expressed by:

$$\frac{\partial F(\mathbf{w})}{\partial w_{ij}} = \frac{\partial h}{\partial \mathbf{f}(\mathbf{w})} \frac{\partial \mathbf{f}(\mathbf{w})}{\partial w_{ij}} \tag{A.5}$$

For $\frac{\partial h(\mathbf{f})}{\partial \mathbf{f}} = \begin{bmatrix} \frac{\partial h}{\partial f_1} & \frac{\partial h}{\partial f_2} & \cdots & \frac{\partial h}{\partial f_m} \end{bmatrix}$ and $\frac{\partial \mathbf{f}(\mathbf{w})}{\partial w_{ij}} = [\frac{\partial f_1}{\partial w_{ij}} \quad \frac{\partial f_2}{\partial w_{ij}} \quad \cdots \quad \frac{\partial f_m}{\partial w_{ij}}]^{\mathrm{T}}$, Expression (A.5) becomes

$$\frac{\partial F(\mathbf{w})}{\partial w_{ij}} = \sum_{k=1}^{m} \frac{\partial h}{\partial f_k(\mathbf{w})} \frac{\partial f_k(\mathbf{w})}{\partial w_{ij}} . \tag{A.6}$$

When i =1,…,n; j=1,…,l, (A.6) becomes:

$$\frac{\partial F(\mathbf{w})}{\partial \mathbf{w}} = \sum_{k=1}^{m} \frac{\partial h}{\partial f_k(\mathbf{w})} \frac{\partial f_k(\mathbf{w})}{\partial \mathbf{w}} . \tag{A.7}$$

7. References

Cichocki, A. & Unbehauen, R. (1993). *Neural Networks for Optimisation and Signal Processing*, New York: John Willey and Sons.

Chryssoloiuris, G., Lee, M., and Ramsey, A., (1996), Confidence Interval Prediction for Neural Network Models, *IEEE Trans. Neur. Networks*, 1, 229-232

Hagan, M.T. & Menhaj, M.B. (1994). Training feedforward networks with the Marquardt Algorithm, *IEEE Trans. Neur. Networks*, 5, 989-993

He. S & Li, J. (2008). Modeling Nonlinear Elastic Behavior of Reinforced Soil Using Artificial Neural Networks, *Applied Soft Computing*, 9(3), 954-961.

He. S & Li, J. (2006). Parameter Estimation of Reinforced Soil Based on Neural Networks, in: Proc. Int. Conf. on Computational Intelligence for Modeling, Control and Automation Sydney, Australia, 137-143.

Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximations, *Neur. Networks*, 2, 359-366.

Hwang, J.T.G. & Ding, A. A. (1997) Prediction intervals for artificial neural networks. J. American Statistical Association 92(438), 748-757.

Issa, R. & Zaman, M. (1999). Estimating resilient modulus of aggregate base by backpropagation neural network model, in: *Proc. 4th Int. Conf. on Constitutive Laws for Engineering Materials*, New York, July, 493-496.

Li, J. & Ding, D. W. (2002). Nonlinear elastic behavior of fiber-reinforced soil under cycle loading, *J. of Soil Dynamics and Earthquake Engineering*, 22 (9) (2002) 265-271.

Li, J. & Zhang, L.J. (2003). Nonlinear elastic behavior of soil reinforced with fiber and lime, in: Proc. 12th Pan-American Conf. on Soil Mechanics and Geotechnical Engineering (eds. P. J. Culligan,, H. H. Einstein, and A. J. Whittle), 610-618.

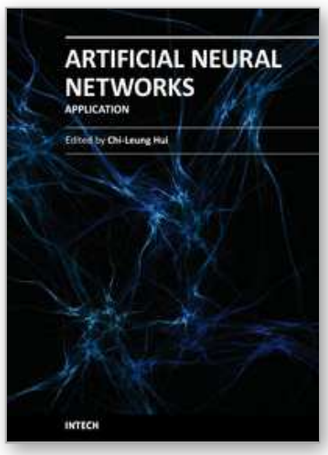
Lewis, F. L. (1995). *Optimal Control*, New York: Wiley-Interscience.

Michalowiski, R. L. & Zhao, A. (1995). Continuum versus structural approach to stability of reinforced soil, *J. of Geotechnical Engineering ASCE*, 121 (2) 152-162.

Rajasekaran, S. & Amalraj, R. (2002). Prediction of design parameter in civil engineering problems using SLNN with a single hidden RBF neuron, *Computers & Structure*, 80 (31) 2495-2505.

Seber, G.A.F. & Wild, C.J. (1989). *Nonlinear Regression*, New York, John Wiley and Sons.

Yang, L., Kavli, T., Carlin, M., Clausen, S, & Groot, P. (2002). An evaluation of confidence bound estimation methods for neural networks, in *Advances in Computational Intelligence and Learning: Methods and Applications*, Kluwer Academic Publishers.



Artificial Neural Networks - Application

Edited by Dr. Chi Leung Patrick Hui

ISBN 978-953-307-188-6
Hard cover, 586 pages
Publisher InTech
Published online 11, April, 2011
Published in print edition April, 2011

This book covers 27 articles in the applications of artificial neural networks (ANN) in various disciplines which includes business, chemical technology, computing, engineering, environmental science, science and nanotechnology. They modeled the ANN with verification in different areas. They demonstrated that the ANN is very useful model and the ANN could be applied in problem solving and machine learning. This book is suitable for all professionals and scientists in understanding how ANN is applied in various areas.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen