ELSEVIER

0954-1810(94)00011-5

Artificial Intelligence in Engineering 9 (1995) 143–151
1995 Elsevier Science Limited
Printed in Great Britain.
0954-1810/95/$09.50

Back-propagation neural networks for modeling complex systems

A. T. C. Goh

School of Civil & Structural Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 2263

In complex engineering systems, empirical relationships are often employed to estimate design parameters and engineering properties. A complex domain is characterized by a number of interacting factors and their relationships are, in general, not precisely known. In addition, the data associated with these parameters are usually incomplete or erroneous (noisy). The development of these empirical relationships is a formidable task requiring sophisticated modeling techniques as well as human intuition and experience. This paper demonstrates the use of back-propagation neural networks to alleviate this problem. Back-propagation neural networks are a product of artificial intelligence research. First, an overview of the neural network methodology is presented. This is followed by some practical guidelines for implementing back-propagation neural networks. Two examples are then presented to demonstrate the potential of this approach for capturing nonlinear interactions between variables in complex engineering systems.

Keywords: back propagation, complex systems, cone penetration test, geotechnical engineering, modeling, neural networks, pile driving.

1 INTRODUCTION

In complex engineering systems, empirical relationships are often employed to estimate design parameters and engineering properties. Generally, a complex domain is characterized by a number of interacting factors in which the relationship between these factors is not precisely known. In addition, the data associated with these parameters are usually incomplete or erroneous (noisy). The extraction of knowledge from the data to develop these empirical relationships is a formidable task requiring sophisticated modeling techniques as well as human intuition and experience.

This paper demonstrates the use of back-propagation neural networks to alleviate this problem. The back-propagation neural network is a product of artificial intelligence research. The growing interest in neural networks among researchers is due to its excellent performance in pattern recognition and the modeling of nonlinear relationships involving a multitude of variables, in place of conventional techniques.

First, an overview of the neural network methodology is presented. This is followed by some practical guidelines for implementing back-propagation neural networks. Two practical examples in geotechnical engineering are then presented to demonstrate the potential of this approach for capturing nonlinear interactions between variables in complex engineering systems. The first example involves the analysis of data obtained from large scale laboratory tests on sand. The other example relates to the prediction of the ultimate load capacity of driven piles from actual field records.

2 NEURAL NETWORKS

2.1 Architecture

A neural network is a computer model whose architecture essentially mimics the knowledge acquisition and organizational skills of the human brain. A thorough treatment of neural network methodology is beyond the scope of this paper. The basic architecture of neural networks has been covered widely.[1,2] A neural network consists of a number of interconnected processing elements, commonly referred to as neurons. The neurons are logically arranged into two or more layers as shown in Fig. 1, and interact with each other via weighted connections. These scalar weights determine the nature and strength of the influence between

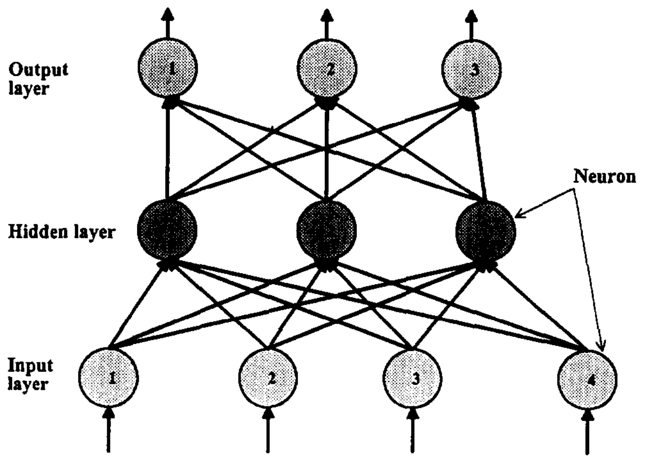


Fig. 1. A typical neural network architecture.

the interconnected neurons. Each neuron is connected to all the neurons in the next layer. There is an input layer where data are presented to the neural network, and an output layer that holds the response of the network to the input. It is the intermediate layers, also known as hidden layers, that enable these networks to represent and compute complicated associations between patterns.

Neural networks essentially learn through the adaptation of their connection weights. A number of neural network learning strategies have been developed and are described elsewhere.[1,2] In recent years, a number of neural network development tools capable of implementing these learning strategies have become commercially available.[3] Some recent applications of neural networks in civil engineering include material modeling,[4] damage assessment,[5,6] structural analysis and design,[7-10] and seismic liquefaction assessment.[11]

2.2 Back-propagation algorithm

The neural network paradigm adopted in this study utilizes the back-propagation learning algorithm.[12] Back-propagation neural networks with a single hidden layer have been shown to be capable of providing an accurate approximation of any continuous function provided there are sufficient hidden neurons.[13] In back-propagation neural networks, the mathematical relationships between the various variables are not specified. Instead, they learn from the examples fed to them. In addition, they can generalize correct responses that only broadly resemble the data in the learning phase.

The basic mathematical concepts of the back-propagation algorithm are found in the literature.[14,15] Training of the neural network is essentially carried out through the presentation of a series of example patterns of associated input and target (expected) output values. Each hidden and output neuron processes its inputs by multiplying each input by its weight, summing the product and then passing the sum through a nonlinear transfer function to produce a result. The S-shaped sigmoid curve is commonly used as the transfer function. The neural network learns by modifying the weights of the neurons in response to the errors between the actual output values and the target output values. This is carried out through the gradient descent on the sum of squares of the errors for all the training patterns.[12] The changes in weights are in proportion to the negative of the derivative of the error term. One pass through the set of training patterns along with the updating of the weights is called a cycle or epoch. Training is carried out by repeatedly presenting the entire set of training patterns (with the weights updated at the end of each cycle) until the average sum squared error over all the training patterns are minimized and within the tolerance specified for the problem. Details of the algorithm for adjusting the weights to minimize the average sum squared error are described in Caudill and Butler[14] and Eberhart and Dobbins.[15]

At the end of the training phase, the neural network should correctly reproduce the target output values for the training data provided the errors are minimal, i.e. convergence occurs. The associated trained weights of the neurons are then stored in the neural network memory. In the next phase, the trained neural network is fed a separate set of data. In this testing phase, the neural network predictions (using the trained weights) are compared with the target output values. This assesses the reliability of the neural network to generalize correct responses for the testing patterns that only broadly resemble the data in the training set. No additional learning or weight adjustments occur during this phase. Once the training and testing phases are found to be successful, the neural network can then be put to use in practical applications. The neural network will produce almost instantaneous results of the output for the practical inputs provided. The predictions should be reliable provided the input values are within the range used in the training set.

3 BACK-PROPAGATION IMPLEMENTATION STRATEGIES

The development of a backpropagation neural network model essentially involves a number of stages. First, the variables to be used as the input parameters for the neural network model have to be identified. This requires an understanding of the problem domain and may require insights from specialists in that field. To minimize the number of input parameters, statistical methods are sometimes used to identify the most significant variables in the model.[16]

The next stage involves gathering the data for use in training and testing the neural network. This requires a data set of case records containing the input patterns and the expected (target output) solution. The training set must provide a representative sample of the data containing the various distinct characteristics of the problem the neural network is likely to encounter in the finished application. A large training set reduces the risk of undersampling the nonlinear function but increases the training time. A general guide is to have at least five to ten training patterns for each weight.[3] As neural networks learn linear relationships more efficiently, to reduce training time, 'one goal of data preparation is to reduce nonlinearity when we know its character and leave the hidden nonlinearities we don't understand for the neural network to resolve'.[17] Hence if it is known that input X is inversely related to the output, a more efficient approach would be to use $(1/X)$ as the input.

Preprocessing of the data is usually required before presenting the patterns to the neural network. This is necessary because the sigmoid transfer function modulates the output of each neuron to values between 0 and 1. Various normalization or scaling strategies have been proposed.[16-18] The following normalization procedure is commonly adopted and was used in this study.

For a variable with maximum and minimum values of V_{max} and V_{min} respectively, each value V is scaled to its normalized value A using

$$A = (V - V_{\mathrm{min}})/(V_{\mathrm{max}} - V_{\mathrm{min}}) \tag{1}$$

The data are then randomly separated into a training set and a testing set. Usually about one-third of the data are used as the testing set.[3] Initially, random scalar weights are assigned to the neurons. The neural network is then fed the training patterns and learns through the adjustment of the weights. Training is carried out iteratively until the average sum squared error over all the training patterns is minimized.

There is currently no rule for determining the optimal number of neurons in the hidden layer except through experimentation. Using too few neurons impairs the neural network and prevents the correct mapping of input to output. Using too many neurons impedes generalization and increases training time.[19] A common strategy and the one used in this study was to replicate the training several times, starting with two neurons and then increasing the number while monitoring the average sum squared error. Training is carried out until there is no significant improvement in the error.

As described earlier, the testing set of patterns is then used to verify the performance of the neural network, on the satisfactory completion of the training. The testing phase assesses the quality of the neural network model and determines whether the neural network can generalize correct responses for patterns that only broadly resemble the data in the training set.

4 NEURAL NETWORK PROGRAM

The back-propagation neural network program adopted in this study essentially followed the formulations of Eberhart and Dobbins.[15] The program was written in C language. Training was carried out until the average sum squared error over all the training patterns was minimized. This occurred after about 10 000–20 000 cycles of training. Training time on a 80486-33 MHz personal computer was usually between 5 and 10 min.

5 EXAMPLE APPLICATIONS

Now two examples are presented to demonstrate the potential of this approach for capturing nonlinear interactions between various parameters in complex civil engineering systems. The first example involves the analysis of data obtained from calibration chamber tests on sand. The other example relates to the prediction of the ultimate load capacity of driven piles. In both examples, actual field data were used in training the neural network. For brevity, only samples of the training and testing data have been included.

5.1 Example A

Cone penetration test (CPT) measurements are often used to determine the soil engineering parameters for use in foundation design. The relationship between the measured cone stresses and the soil properties are determined from empirical correlations. In sands, these correlations are commonly derived from large scale laboratory calibration chamber tests. The sand sample of known density is prepared in the chamber and then consolidated to the desired stresses. The cone is then pushed into the sample, and the cone tip resistance q_c and the sleeve friction f_s are measured. The engineering properties of the sample are determined from laboratory testing. The cone measurements are then correlated directly to the engineering properties.

CPT calibration chamber tests have been carried out by a number of researchers including Holden[20] and Baldi *et al.*[21] For this study, the experimental results obtained by Baldi *et al.*[21] were used. Their experiments involved a comprehensive study of the behaviour and properties of Ticino sand, under different stress and boundary conditions. From statistical analysis, they were able to establish correlations between q_c and a number of engineering parameters. In this paper, the correlation between the tangent constrained modulus M_o during compression and q_c for normally consolidated sand is considered. The correlation determined by Baldi *et al.*[21] from statistical analysis is shown below.

$$M_o/q_c = 1420(\sigma'_m/98{\cdot}1)^{-0{\cdot}116}e^{-1{\cdot}123D_R} \tag{2}$$

The mean effective stress σ'_m, M_o and q_c are in units of kPa, and the sand relative density D_R is in decimals. This statistical correlation is used for comparison with the neural network predictions.

Table 1. Sample training and testing data for Example A

D_R (%)	σ'_m (kPa)	q_c (MPa)	M_o (MPa)
Training			
92·4	366·4	46·5	147·6
92·9	221·2	39·1	119·6
92·9	80·5	23·9	80·3
74·6	221·3	26·1	106·1
74·6	369·1	34·4	131·4
74·9	516·3	40·7	144·3
61·8	370·2	20·1	111·4
63·4	515·3	25	120·5
91·8	46·7	18·4	71·1
75·8	46·4	10·9	66·3
Testing			
73·1	80·6	15·6	74·3
92·9	219·8	36·2	118·1
57·7	223·1	13·4	87·5
61·8	81·6	9·1	62·6
63·4	45·4	5·6	52·1
55·8	370·1	15·5	112·2
76·7	224·8	22·1	108·4
56·4	522·1	19·9	128·4
77·2	518·8	32·1	137·7
51·2	85	7·3	60·8

The neural network consisted of three input neurons representing D_R, σ'_m and q_c and a single output neuron representing M_o. The D_R values ranged from 16% to 96%, σ'_m ranged from 26 to 458 kPa, q_c was in the range 2–47 MPa, and M_o was in the range 16–150 MPa. A total of 73 training patterns and 29 testing patterns were used. Some sample training and testing patterns are shown in Table 1.

The average sum squared error plotted as a function of the training cycles is shown in Fig. 2 for the neural network with four hidden neurons. Experiments indicated that there was no significant improvement in convergence as the number of hidden neurons increased beyond four. The neural network predictions for the training and testing sets are shown in Fig. 3. The scatter of the predicted M_o values versus the measured M_o values were assessed using regression analysis. High coefficients of correlations for the training and testing data were obtained as shown in Table 2. The results show that the neural network was successful in modeling the nonlinear relationship between M_o and the other parameters. A comparison of the correlation coefficients in Table 2 indicates that neural network model is more reliable than the statistical model. This is also evident from the plots in Fig. 3.

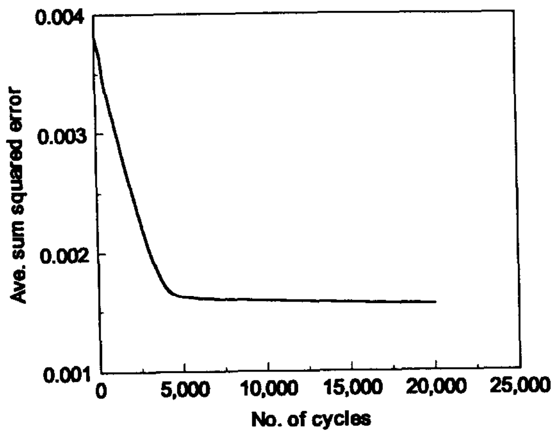


Fig. 2. Convergence characteristics during training for Example A with four hidden neurons.

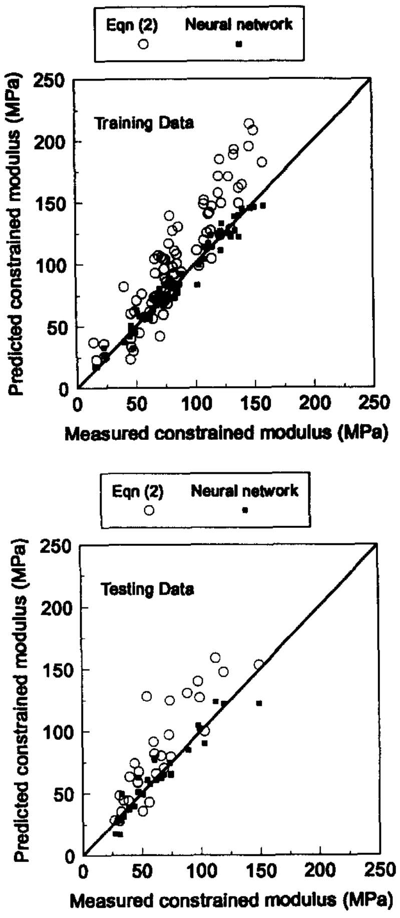


Fig. 3. Comparison of predicted and measured M_o values.

Table 3 shows the weights of the hidden-input layer connections, and the hidden-output layer connections. The relative importance of the various input factors can be assessed by examining these connection weights of the neurons. This involves partitioning the hidden-output connection weights into components connected with each input neuron.[22] The results are summarized in Table 3. An example of the computational process is shown in the Appendix. They indicate that D_R is the

Table 2. Summary of regression analysis results for Example A

Method	Coefficient of correlation	
	Training data	Testing data
Neural network	0·98	0·94
Eqn (2)	0·91	0·87

Table 3. Summary of connection weights for Example A

Hidden neurons	Weights			
	D_R	σ'_m	q_c	M_o
Hidden 1	−1·68	3·29	1·32	4·58
Hidden 2	−0·52	−0·23	−0·26	−0·49
Hidden 3	−4·02	2·12	−0·08	−5·74
Hidden 4	−1·76	−1·45	0·58	−2·65
Relative importance (%)	47·3	36·9	15·8	—

most important input factor, followed by σ'_m, with q_c of less importance.

Since the neural network is capable of generalization, parametric studies can be carried out to evaluate the effects of the various input parameters D_R, σ'_m and q_c on the output M_o. This was done at the end of the testing phase, whereby the trained neural network was fed some hypothetical values of D_R, σ'_m and q_c. The results are shown in Fig. 4 demonstrating the correlations between D_R, σ'_m, q_c and the predicted M_o values.

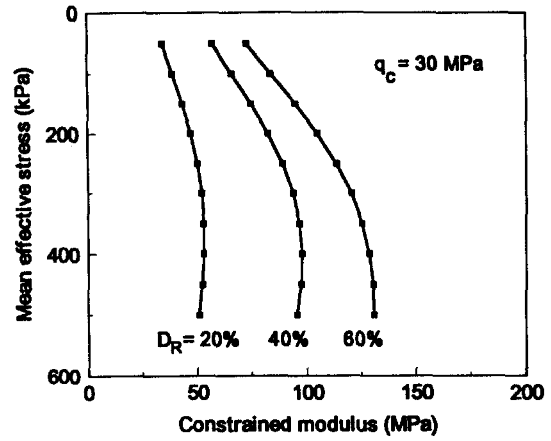


Fig. 4. Results of typical parametric study.

5.2 Example B

The second example involves the estimation of the load capacity of driven piles. Pile driving formulae are commonly used to estimate the load capacity of driven piles. These formulae are essentially derived from impulse-momentum principles. The formulae assume that there is a correlation between the driving resistance and the ultimate load capacity of the pile Q_u. The important factors influencing the load capacity include the hammer characteristics, the properties of the pile and soil, and the pile set s.

A number of pile driving formulae are widely used in actual practice. These include the Engineering News (EN) formula,[23] the Hiley formula,[24] and the Janbu formula.[25] These formulae are summarized in Table 4, where W is the hammer weight, H is the hammer drop, L is the pile length, W_p is the pile weight, A is the pile cross-sectional area, and E is the pile modulus of elasticity. The derivation of these formulae is described by Whitaker,[27] and is beyond the scope of this paper.

The relative merits and reliability of the various conventional formulae have been discussed by others including Olson and Flaate,[28] and will not be covered in this paper.

The training and testing data were drawn from actual case records compiled by Flaate[29] for piles in cohesionless soils. This consisted of the results of load tests on timber, precast concrete and steel piles driven into sandy soils. Details of the range of values for the records are summarized in Table 5. A total of 59 patterns were randomly selected for the training phase and 35 patterns for the testing phase. The output neuron was the pile load capacity Q_u. Some sample training and testing data are shown in Table 6.

Experiments were carried out using a number of combinations of input parameters to determine the most reliable neural network model. Generally the reliability of the model improved as the number of input parameters increased. The neural network model with eight input neurons representing E, L, A, W_p, H, W, s, and the hammer type (H type) and three hidden neurons was found to be the most reliable. H type was assigned a binary value of 1 for gravity hammers and a value of 0 for steam hammers.

The average sum squared error plotted as a function of the training cycles is shown in Fig. 5. The results indicate that convergence was achieved for the training

Table 4. Pile driving formulae

Formula	Equation for Q_u	Remarks
Engineering News (EN)	$\dfrac{WH}{s+c}$	c = 25 mm for gravity hammer = 2·5 mm for steam hammer = 2·5 W_p/W for steam hammer on very heavy piles
Hiley	$\dfrac{e_f WH}{s+0{\cdot}5(c_1+c_2+c_3)}\cdot\dfrac{W+n^2W_p}{W+W_p}$	e_f, c_1, c_2, c_3 and n are tabulated by Chellis[26]
Janbu	$\dfrac{WH}{k_u s}$	$k_u = C_d\{1+(1+\lambda_e/C_d)^{0{\cdot}5}\}$ $\lambda_e = WHL/AEs^2$ $C_d = 0{\cdot}75 + 0{\cdot}15\,W_p/W$

Table 5. Summary of range of values for Example B

Property	Symbol	Range of values
Pile elastic modulus (GPa)	E	9·7–206·8
Pile length (m)	L	4·4–32·4
Pile cross-sectional area (m^2)	A	$0·8 \times 10^{-3}$ to 0·37
Pile weight (kN)	W_p	0·7–221
Hammer drop (m)	H	0·5–4·1
Hammer weight (kN)	W	0·98–48·9
Pile set (mm)	s	0·76–76·2
Hammer type	Htype	Gravity or steam hammer

phase. The neural network predictions for the training and testing sets are shown in Fig. 6. The scatter of the predicted Q_u values versus the measured Q_u values were assessed using regression analysis. High coefficients of correlations for the training and testing data were obtained as shown in Table 7. The results from the testing phase suggest that although the model was not explicitly trained for these data, the neural network was capable of generalization and generally gave reasonable predictions. The results indicate that the neural network was successful in modeling the nonlinear relationship between Q_u and the other parameters.

The neural network results in Fig. 6 showed less scatter in the data points in comparison to the conventional methods listed in Table 4. For brevity, the plots of the measured and predicted Q_u using these conventional methods have been omitted. The coefficients of correlation of predicted versus measured results are summarized in Table 7. They indicate that the neural network predictions are more reliable than the conventional pile driving formulae.

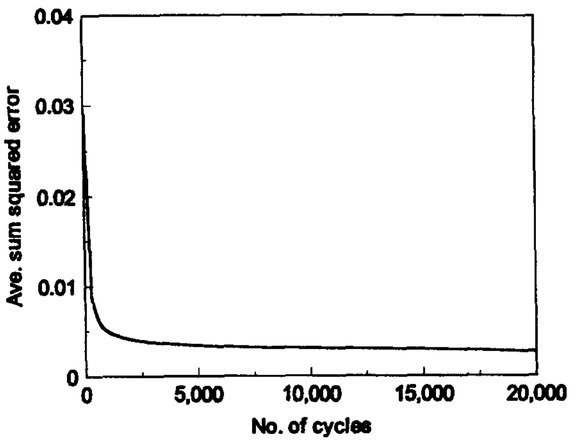


Fig. 5. Convergence characteristics during training for Example B with three hidden neurons.

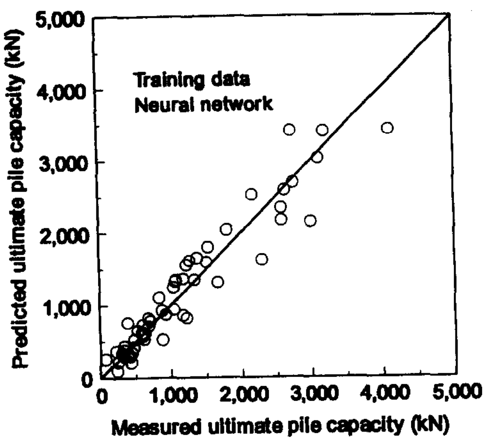
6 DISCUSSION

Statistical methods are commonly used in the development of empirical relationships between various interacting factors. This is often complex and circuitous, particularly for nonlinear relationships. Also, to formulate the statistical model, the important parameters must be known. By comparison, the modeling process in back-propagation neural networks is more direct, as there is no necessity to specify a mathematical

Table 6. Sample training and testing data for Example B

E (GPa)	L (m)	A ($\times 10^{-3} m^2$)	W_p (kN)	H (m)	W (kN)	s (mm)	H type[a]	Q_u (kN)
Training								
9·7	13·61	70·33	6·85	1·02	10·76	19·3	1	302·46
9·7	9·7	50·65	4·89	0·74	13·34	9·65	0	329·15
17·9	9·3	130·98	26·87	0·61	17·7	3·56	1	1 076·42
206·8	32	4·59	16·01	0·91	22·24	0·76	0	1 272·13
206·8	11·79	8·26	10·23	0·76	13·34	3·25	0	880·7
206·8	17·65	15·48	19·57	0·81	39·14	1·83	1	2 757·76
206·8	17·78	33·42	43·06	0·81	39·14	6·1	1	1 805·89
9·7	13·61	70·33	6·85	1·02	10·76	22·35	1	311·36
9·7	7·16	72·91	6·85	1·12	26·42	14·22	1	934·08
9·7	24	232·27	44·04	1·52	39·14	8·13	1	1 663·55
Testing								
9·7	19·3	187·11	26·69	1·02	39·14	2·03	1	1 094·21
206·8	17·65	15·48	19·57	0·81	39·14	2·24	1	2 633·22
9·7	19·81	111·62	14·95	1·02	29·45	12·19	1	951·87
9·7	9·3	46·45	3·38	0·64	13·34	8·64	0	329·15
9·7	16·26	82·59	9·79	0·71	12·72	1·52	1	871·81
206·8	23·37	14·65	28·38	1·02	39·14	12·19	1	1 103·1
206·8	17·65	12·9	18·59	1·02	17·61	1·93	1	2 072·77
9·7	23·88	187·11	33·27	1·02	39·14	10·16	1	1 076·42
206·8	22·35	13·42	21·53	1·02	39·14	8·13	1	1 975·12
17·9	24·74	370·99	182·63	0·99	41·37	7·62	0	1 227·65

[a] Gravity hammer = 1, steam hammer = 0.



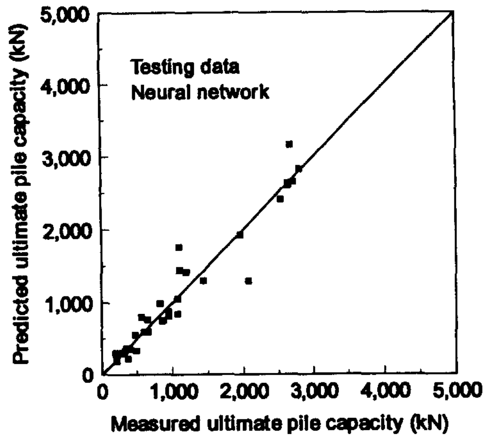


Fig. 6. Comparison of predicted and measured Q_u values.

relationship between the input and output variables. Neural networks can be effective for analysing a system containing a number of variables, to establish patterns and characteristics not previously known. In addition, it can generalize correct responses that only broadly resemble the data in the training set. During training, irrelevant input variables are assigned low connection weights. These variables can then be omitted from the model. In neural networks, quantitative as well as qualitative information can be considered. This was illustrated in Example B where qualitative information related to the hammer type Htype was incorporated into the model. Since the neural networks are trained on actual test data, they are trained to deal with inherent noisy or imprecise data. As new data become available, the neural network model can be readily updated by retraining with patterns which include these new data.

Table 7. Summary of regression analysis results for Example B

Method	Coefficient of correlation	
	Training data	Testing data
Neural network	0·96	0·97
Engineering News (EN)	0·69	0·61
Hiley	0·48	0·76
Janbu	0·82	0·89

The main criticism of the neural network methodology is its inability at present to trace and explain the step-by-step logic it uses to arrive at the outputs from the inputs provided. This is expected to be a temporary drawback that will be overcome with further research.

7 SUMMARY

This study demonstrates the feasibility of using neural networks for capturing nonlinear interactions between various parameters in complex civil engineering systems. A simple back-propagation neural network was used to model two problems involving nonlinear variables. Actual field data were used. After learning from a set of selected patterns, the neural network models were able to produce reasonably accurate predictions.

REFERENCES

1. Rumelhart, D. E. & McClelland, J. L. *Parallel Distributed Processing — Explorations in the Microstructure of Cognition*, Vols 1 and 2. MIT Press, Cambridge, MA, 1986.
2. Lippmann, R. P., An introduction to computing with neural nets. *IEEE Acoust. Speech Signal Process,* **4**(2) (1987) 4–22.
3. Hammerstrom, D. Working with neural networks. *IEEE Spectrum,* July (1993) 46–53.
4. Ghaboussi, J., Garrett, J. H. Jr & Wu, X. Knowledge-based modeling of material behavior with neural networks. *J. Engng. Mech. Division ASCE,* **117**(1) (1991) 132–53.
5. Elkordy, M. F., Chang, K. C. & Lee, G. C. Neural networks trained by analytically simulated damage states. *J. Comput. Civil Engng ASCE,* **7**(2) (1993) 130–45.
6. Yeh, Y. C., Kuo, Y. H. & Hsu, D. S. Building KBES for diagnosing PC pile with artificial neural network. *J. Comput Civil Engng ASCE,* **7**(1) (1993) 71–93.
7. Vanluchene, D. & Sun, R. Neural networks in structural engineering. *Microcomp. Civil Engng,* **5**(3) (1990) 207–15.
8. Hajela, P. & Berke, L. Neurobiological computational models in structural analysis and design. *Comput. Struct.,* **41**(4) (1991) 657–67.
9. Hajela, P., Fu, B. & Berke, L. Neural networks in structural analysis and design: an overview. *Comput. Syst. Engng,* **3**(1–4) (1992) 525–38.
10. Gunaratnam, D. J. & Gero, J. S. Effect of representation on the performance of neural networks in structural engineering applications. *Microcomp. Civil Engng,* **9**(2) (1994) 97–108.
11. Goh, A. T. C. Seismic liquefaction potential assessed by neural networks. *J. Geot. Engng ASCE,* **120**(9) (1994) 1467–80.
12. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning internal representation by error propagation. In *Parallel Distributed Processing,* ed. R. H. Rumelhart & J. L. McClelland. MIT Press, Cambridge, MA, 1986.
13. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks,* **4**(2) (1991) 251–7.
14. Caudill, M. & Butler, C. *Naturally Intelligent Systems.* MIT Press, Cambridge, MA, 1990.
15. Eberhart, R. C. & Dobbins, R. W. *Neural Network PC*

Tools: A Practical Guide. Academic Press, San Diego, 1990.
16. Stein, R. Selecting data for neural networks. *AI Expert*, **8**(2) (1993) 42–7.
17. Crooks, T. Care and feeding of neural networks. *AI Expert*, **7**(9) (1992) 36–41.
18. Masters, T. *Practical Neural Network Recipes in C++*. Academic Press, San Diego, 1993.
19. Bailey, D. & Thompson, D. How to develop neural network applications. *AI Expert*, **5**(6) (1990) 38–47.
20. Holden, J. The calibration of electrical penetrometers in sand. Internal report, Norwegian Geotechnical Institute, Oslo, 152108-2, 1976.
21. Baldi, G., Bellotti, R., Ghionna, V. N., Jamiolkowski, M. & Pasqualini, E. Interpretation of CPTs and CPTUs — 2nd Part: Drained penetration of sands. *Proc. 4th Int. Geotech. Seminar on Field Instrumentation and Insitu Measurements*. Nanyang Technological Institute, Singapore, 1986, pp. 143–56.
22. Garson, G. D. Interpreting neural-network connection weights. *AI Expert*, **6**(7) (1991) 47–51.
23. Wellington, A. M. The iron wharf at Fort Monroe, VA. *Trans. ASCE*, **27** (1892) 129–37.
24. Hiley, A. The efficiency of the hammer blow, and its effects with reference to piling. *Engineering*, 2 June (1922) 673.
25. Janbu, N. Une analyse energetique du battage des pieux a l'aide de parametres sans dimension. Norwegian Geotechnical Institute, Oslo, 1953, pp. 63–4 (in Norwegian).
26. Chellis, R. D. *Pile Foundations*, 2nd edn. McGraw-Hill, New York, 1961.
27. Whitaker, T. *The Design of Piled Foundations*. Pergamon Press, Oxford, 1970.
28. Olson, R. E. & Flaate, K. S. Pile-driving formulas for friction piles in sand. *J. Soil Mech. Foundat. Div. ASCE*, **93**(6) (1967) 279–96.
29. Flaate, K. S. An investigation of the validity of three pile driving formulae in cohensionless material. Norwegian Geotechnical Institute, Oslo, 1964, pp 11–22.

APPENDIX. EXAMPLE ILLUSTRATING THE PARTITIONING OF WEIGHTS

This appendix details the procedure for partitioning the connection weights to determine the relative importance of the various inputs, using the method proposed by Garson.[22] The method essentially involves partitioning the hidden-output connection weights of each hidden neuron into components associated with each input neuron.

Consider the neural network with three input neurons, four hidden neurons and one output neuron with the connection weights as shown below, as an example.

Hidden neurons	Weights			
	Input 1	Input 2	Input 3	Output
Hidden 1	−1·67624	3·29022	1·32466	4·57857
Hidden 2	−0·51874	−0·22921	−0·25526	−0·48815
Hidden 3	−4·01764	2·12486	−0·08168	−5·73901
Hidden 4	−1·75691	−1·44702	0·58286	−2·65221

The computation process is as follows:

(1) For each hidden neuron i, multiply the absolute value of the hidden-output layer connection weight by the absolute value of the hidden-input layer connection weight. Do this for each input variable j. The following products P_{ij} are obtained:

	Input 1	Input 2	Input 3
Hidden 1	$P_{11} = 1{\cdot}67624 \times 4{\cdot}57857$	$P_{12} = 3{\cdot}29022 \times 4{\cdot}57857$	$P_{13} = 1{\cdot}32466 \times 4{\cdot}57857$
Hidden 2	$P_{21} = 0{\cdot}51874 \times 0{\cdot}48815$	$P_{22} = 0{\cdot}22921 \times 0{\cdot}48815$	$P_{23} = 0{\cdot}25526 \times 0{\cdot}48815$
Hidden 3	$P_{31} = 4{\cdot}01764 \times 5{\cdot}73901$	$P_{32} = 2{\cdot}12486 \times 5{\cdot}73901$	$P_{33} = 0{\cdot}08168 \times 5{\cdot}73901$
Hidden 4	$P_{41} = 1{\cdot}75691 \times 2{\cdot}65221$	$P_{42} = 1{\cdot}44702 \times 2{\cdot}65221$	$P_{43} = 0{\cdot}58286 \times 2{\cdot}65221$

(2) For each hidden neuron, divide P_{ij} by the sum for all the input variables to obtain Q_{ij}. For example for Hidden 1, $Q_{11} = P_{11}/(P_{11} + P_{12} + P_{13}) = 0{\cdot}266445$.

(3) For each input neuron, sum the product S_j formed from the previous computations of Q_{ij}. For example, $S_1 = Q_{11} + Q_{21} + Q_{31} + Q_{41}$.

	Input 1	Input 2	Input 3
Hidden 1	$Q_{11} = 0{\cdot}266445$	$Q_{12} = 0{\cdot}522994$	$Q_{13} = 0{\cdot}210560$
Hidden 2	$Q_{21} = 0{\cdot}517081$	$Q_{22} = 0{\cdot}228478$	$Q_{23} = 0{\cdot}254441$
Hidden 3	$Q_{31} = 0{\cdot}645489$	$Q_{32} = 0{\cdot}341388$	$Q_{33} = 0{\cdot}013123$
Hidden 4	$Q_{41} = 0{\cdot}463958$	$Q_{42} = 0{\cdot}382123$	$Q_{43} = 0{\cdot}153919$
Sum	$S_1 = 1{\cdot}892973$	$S_2 = 1{\cdot}474983$	$S_3 = 0{\cdot}632044$

(4) Divide S_j by the sum for all the input variables. Expressed as a percentage, this gives the relative importance or distribution of all output weights attributable to the given input variable. For example, for the input neuron 1, the relative importance is equal to $(S_1 \times 100)/(S_1 + S_2 + S_3) = 47{\cdot}3\%$.

	Input 1	Input 2	Input 3
Relative importance (%)	47·3	36·9	15·8