AICTE-ISTE Sponsored Short Term Training Programme

on

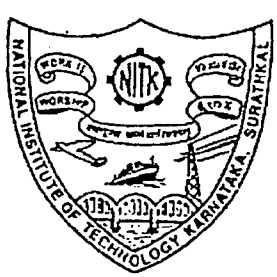
Coastal Erosion Areas (CEA) – Protection & Management

06 – 18, January , 2003

LECTURE VOLUME

Dr. A. Vittal Hegde
Co-ordinator

Dr. Subba Rao
Co-ordinator



Department of Applied Mechanics & Hydraulics
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
(A Deemed University)
Surathkal, Srinivasnagar – 575 025, D.K. Karnataka, India

APPLICATION OF NEURAL NETWORKS IN COASTAL ENGINEERING

Dr. S. Mandal
Assistant Director
Ocean Engineering Division
National Institute of Oceanography
Dona Paula, Goa 403 004
Email: smandal@darya.nio.org

INTRODUCTION

Neural networks have the ability to recognize the hidden pattern in the data and accordingly estimate the output values. Provision of model-free solutions, data error tolerance, built in dynamism and lack of any exogenous input requirement makes the neural network attractive. A neural network is an information processing system modeled on the structure of the dynamic process. It can solve the complex/nonlinear problems quickly once trained by operating on problems using an interconnected number of processing units. Its merit is the ability to deal with information whose interrelation is ambiguous or whose functional relation is not clear.

Basically we need actual input-output data of any particular process/system which will be used to train the neural network. Once the network is trained, it estimates/predicts the output for unknown input data without any assumptions. In many occasions, neural network predictions are much closer to target output and better than other numerical / empirical methods. That is why it is becoming popular in various fields including coastal engineering.

Kingston et al (2000) describes an artificial neural network to model the cross-shore movement in the intensity maximum due to tides and waves. The resulting signal is then removed from the video data estimates to provide an accurate estimation of the sandbar location. Recent work on neural networking for tidal prediction is carried out by Deo and Chaudhuri (1998), Mandal (2001), Lee and Jeng (2002). Similarly, Deo et al(1999, 2002), Subbarao et al (2001), Tsai et al (2002), Mandal et al (2002) have carried out the wave prediction using neural networks.

Waves and tides will play important roles in coastal erosion or accretion. This paper briefly describes the back-propagation neural networks and its application towards prediction of waves and tides.

NEURAL NETWORK

The word 'Neural Network' is used to normally describe the "Artificial Neural Network"(ANN). Biological neural networks are much more complicated in their elementary structures than the mathematical models, which we use for ANNs. An ANN is a network of many very simple processors called "units", each possibly having a small amount of local memory. The units are connected by unidirectional communication channels called "connections", which carry numeric (as opposed to symbolic) data. The units operate only on their local data and on the inputs they receive through the connections. The easy adaptability to the solution is what distinguishes neural networks from other mathematical techniques.

Feed forward neural networks

Normally in a feed forward network there will be three layers namely the input layer, the hidden layer and the output layer. The inputs to the network are given at input layer. The number of input nodes would be equal to the number of input parameters. So the input nodes receive the data and pass them on to the hidden layer nodes. These nodes individually sum up the received values after multiplying each input value by a weight. Then they attach a bias to this sum and pass on the result through a non-linearity such as the "sigmoid transfer function". This forms the input to the output layer that operates identically with the hidden layer nodes. Resulting transformed output from each output node forms the network output.

In the present work the back-propagation feed forward type network is used. The objective is to minimize the global error, E given as

$$E = 1/P \, \Sigma \, E_p$$

and

$$E_p = 0.5 \, \Sigma \, (o_k - t_k)^2$$

where P = the total number of training patterns

E_p = the error for p^{th} pattern.

o_k = network output at k^{th} output node

t_k = target output at k^{th} output node.

In the back-propagation networks the error between the target output and the network output is calculated and this will be back-propagated using the steepest descent or gradient descent approach. The network weights and biases are adjusted by moving a small step in the direction of negative gradient of the error function during each iteration. The iterations are

epeated until a specified convergence is reached. Due to the fixed step size it converges lowly and may exhibit oscillatory behavior and hence the back-propagation networks with pdated algorithms are used in this paper. A brief description about the working of a back propagation neural network and three updated algorithms is given below.

Back-propagation learning

Back-propagation is the most widely used algorithm for supervised learning with multi layer feed forward networks. The idea of the back-propagation learning algorithm is the repeated application of the chain rule to compute the influence of each weight in the network with respect to an arbitrary error function E:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}}$$

In this w_{ij} is the weight from neuron j to neuron i, s_i is the output, and net_i is the weighted sum of the inputs of neuron i. Once the partial derivative for each weight is known, the aim of minimizing the error function is achieved by following updated algorithms

Quickprop

The Quickprop algorithm is developed from the Newton's method. The method is similar to normal back-propagation, but the variation is that for each weight this process keeps storing the error derivative computed during the previous training epoch, along with the difference between the current and previous values of this weight. The value for the current training epoch is also available at weight update time. (Fahlman 1988). The estimates of the position of the minimum for each weight are obtained by solving the equation given below

$$\Delta w_{ij}(t) = \frac{\frac{\partial E}{\partial w_{ij}}(t)}{\frac{\partial E}{\partial w_{ij}}(t-1) - \frac{\partial E}{\partial w_{ij}}(t)} \Delta w(t-1)$$

In order to achieve the minimum E, the above weight increment is to be added or subtracted to the weight update.

RESULTS

Some results are shown below, which provide usefulness of the neural networks in coastal engineering.

The neural network structure (Fig 1) used in the present work is $I_3H_2O_2$ where I_3 = three input nodes (difference between central and peripheral pressures, radius of maximum wind and speed of forward motion), H_2 = two hidden nodes and O_2 = two output nodes (wave height and wave period). The number of training iterations used in the network structure is varied depending on estimated error values. The initial values of the interconnection weights and thresholds are given by uniform random numbers from -1 to 1. Once the learning process is completed from a training set, the final interconnection weights and thresholds are used in the prediction process. The tracks of the 11 cyclones considered for the present study are shown in Fig2. Out of the above 11 cyclones, 3 cyclone data (- - - as shown in Fig2) are considered to train the neural network. The remaining 8 cyclone data is used for testing the given neural network architecture (Figs 3 & 4). The estimated correlation coefficients are 0.98 and 0.97 for wave height and period respectively.

Similarly, the neural network structure for tidal prediction is considered as $I_6H_2O_1$. Here a previous time series of tides is generally used to predict the next tidal value. The present neural network model uses six previous consecutive values to predict the next one, for example, x_1, x_2, x_3, x_4, x_5, x_6 tidal data are used to predict next tidal value x_7. This is similar to the Autoregressive Moving Average (ARMA) process (Mandal,1996). Fig-5 used 1-day tidal value as training set where final interconnection weights and values of thresholds are calculated and these values are used to predict for next 30-days tides. The comparison between the measured and predicted tides for 30-days is also shown in Fig 5. The estimated correlation coefficient is 0.998. This shows that a very good agreement between the measured and predicted tides can be obtained using the neural network model.

Many coastal engineering problems related to coastal processes which lead to coastal erosion or accretion, are generally solved based on SPM (1984) formulas and simplified inputs with assumptions. Very recently published book by Prof S Narasimhan et al (2002) has highlighted about port and coastal engineering in Indian scenario. The more accurate inputs will provide better estimation of shoreline changes. However, in many occasions, actual data collection in swash and surf zones is not an easy task. Therefore, one has to make some assumptions for sediment transport rate estimation. Whereas, the neural networks can solve complex phenomena without any restrictions.

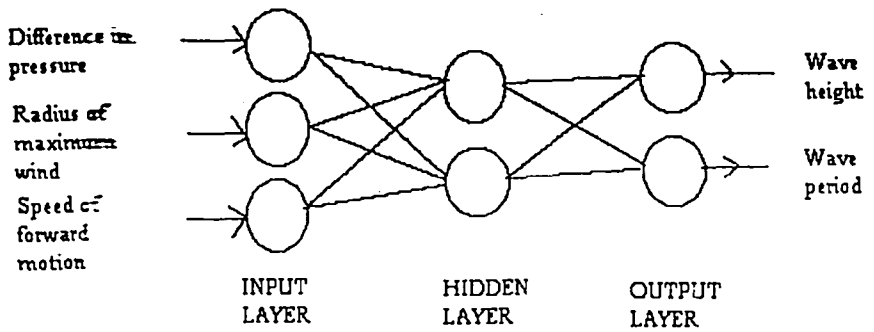


Fig-1 : An NN structure for cyclonic wave hind-casting

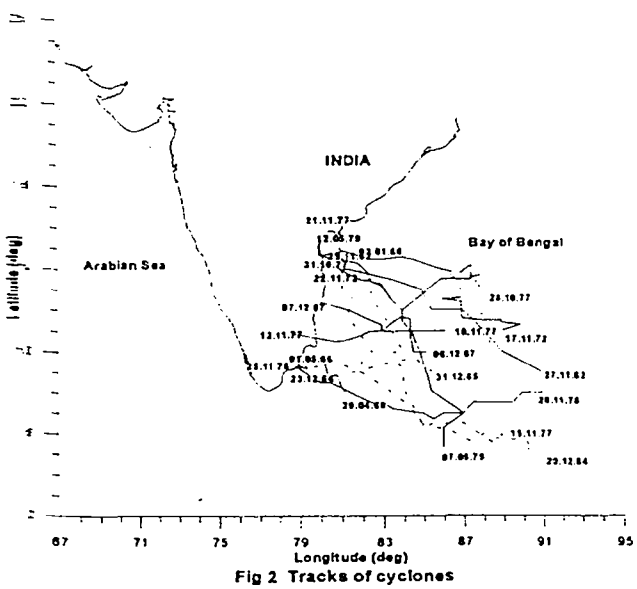


Fig 2 Tracks of cyclones

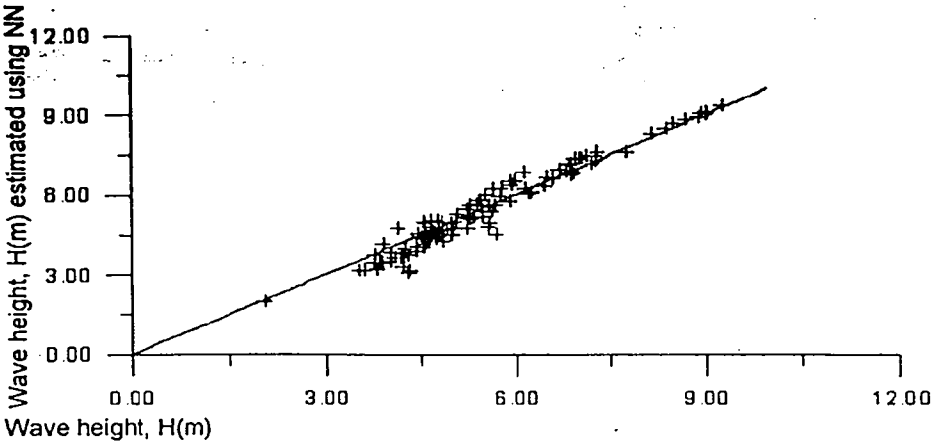


Fig 3 Correlation between wave heights estimated using NN and Young's model (CC=0.98)

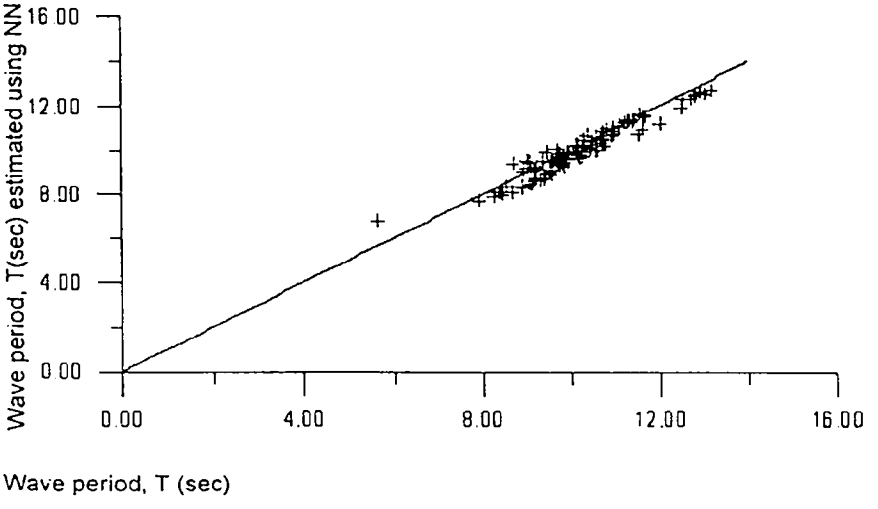


Fig.4 Correlation between spectral peak period estimated using NN and Young's model (CC=0.97)

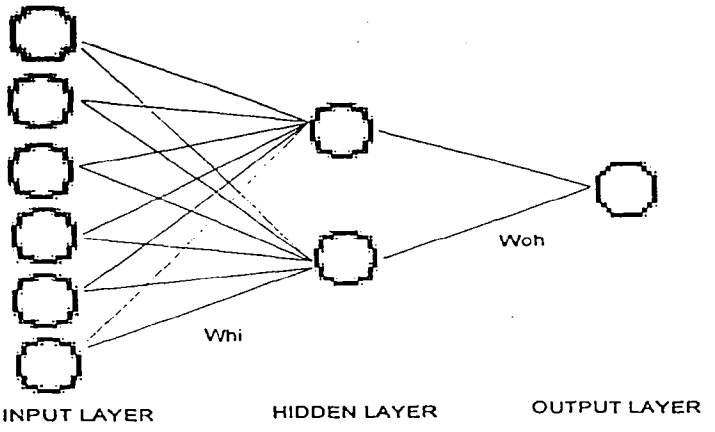


Fig.1 Structure of ANN

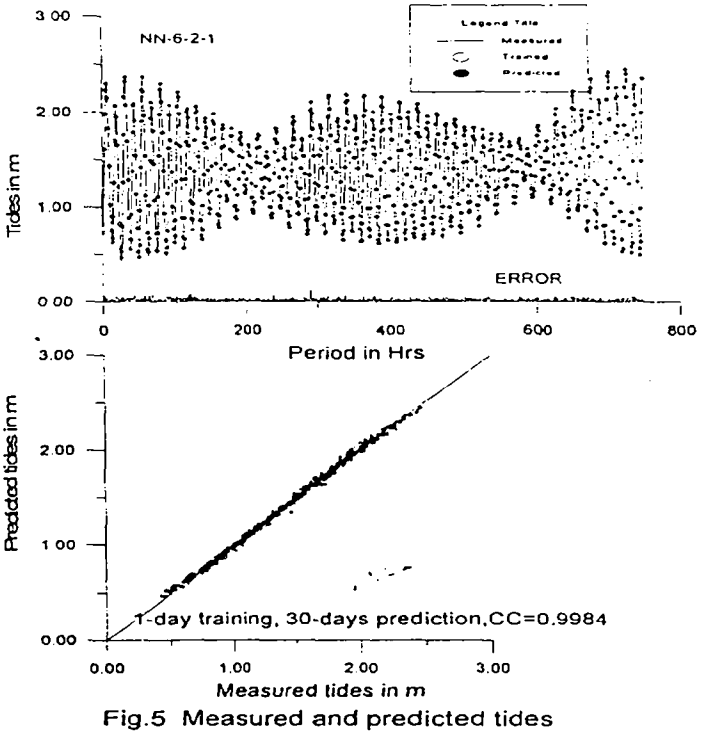


Fig.5 Measured and predicted tides

FUTURE WORKS

The neural networks can be efficiently used:

- To replace sediment transport equations

- To generate beach profiles
- To predict shoreline changes (erosion / accretion)
- To predict grain sizes

REFERENCES

Deo, MC and Chaudhuri,G (1998) Tide prediction using neural networks, J of computer-aided civil and infrastructural engineering, (UK), 13, 113-120.

Deo, MC and Naidu, CS (1999) Real time wave forecasting using neural networks, Ocean Engineering, 26, 191-203.

Deo, MC, Gondane, DS and Kumar, VS (2002) Analysis of wave directional spreading using neural networks. J of Waterway, port, coastal and ocean engineering, 128(1), 30-37.

Fahlman, SE (1988) An emperical study of learning speed in backpropagation networks. Technical report, CMU-CS-88-161, Carnegie-Mellon Univ, Computer Science Dept, Pittsburgh, PA.

Kingston, KS; Ruessink, BG; Enckevort, IMJv and Davidson, MA (2000) Artificial neural network correction of remotely sensed sandbar location. Marine Geology, 169, 137-160.

Lee, TL and Jeng, DS (2002) An application of artificial neural networks in tide-forecasting. Ocean Engineering, 29, pp 1003-1022

Mandal,S; Subba Rao and Chackraborty, NV (2002) Hindcasting cyclonic waves using neural network. International Conference SHOT 2002, IIT Kharagpur, 18-20 December, 2002.

Mandal, S (1996) Optimal parametric modelling of measured short waves. Proc. Intl. Conf in Ocean Engineering, IIT Madras, pp 237-242.

Mandal, S (2001) Tides prediction using backpropagation neural networks, Proc. International Conference in Ocean Engineering, IIT Madras, pp 499-504.

Mandal, S (2001) Back-propagation neural networks in tidal level forecasting, Journal of Waterway, Port, Coastal and Ocean Engineering, January, pp 55.

Narasimhan, S; Kathiroli, S and Nagendra Kumar, B (2002) Harbour and coastal engineering (Indian scenario), NIOT, Chennai.

Reidmiller, M. and Braun, H. (1993) A direct adaptive method for faster backpropagation lerning: The RPROP algorithm. Proc. International Conference in neural networks, San Francisco, CA.

Reidmiller, M. (1994) Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms, Computers Standards and Interfaces, Special issue on neural networks (5).

Subba Rao, Mandal, S and N Prabhaharan Wave forecasting using neural networks, Proc. International Conference in Ocean Engineering, IIT Madras, 2001, pp 103-108.

SPM (1984) Shore protection manual, Coastal Engineering Research Centre, US Department of Army, Corps of Engineers, Washington DC..

Tsai, CP, Lin, C and Shen, JN (2002) Neural network for wave forecasting among multi-stations. Ocean Engineering, 29, pp 1683-1695.

ANNEXURE-I: To solve the xor problem using simple back-propagation neural network

The exlusive-or (xor) is defined as follows

x	y	z
1	0	1
0	0	0
0	1	1
1	1	1

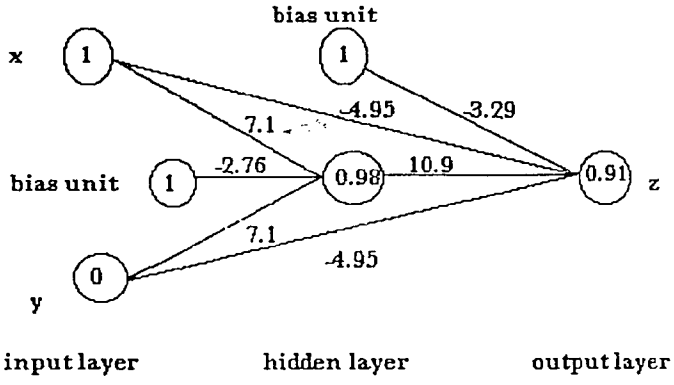


Fig-A1 A three layer network to solve xor problem with weights produced by back- propagation.

Here the circles represent *neurons or nodes or units* and the numbers within the circles represent the activation values of the nodes. The main nodes are arranged in layers. There are three layers: the *input layer* that contains the values for x and y, a *hidden layer* that contains one node, h and an *output layer* that gives the output value, z. The hidden layer is so-named because the network can be regarded as a black box with inputs and outputs that can be seen but the hidden units cannot be seen. There are two other units present called *bias units* whose values are always 1.0. The lines connecting the circles represent *weights* and the number beside a weight is the value of the weight. Much of the time back-propagation networks only have connections within adjacent layers however this one has two extra connections that go directly from the input units to the output unit. In some problems, like xor these extra input-output connections make training the network much faster. Networks are usually just described by the number of units in each layer so the network in Figure-A1 can be described as a 2-1-1 network with extra input-output connections. This will be shortened to 2-1-1-x.

To compute the value of the output unit, z, we place values for x and y on the input layer units. Let these values be 1.0 and 0.0 as in Figure-A1. First we compute the value of the hidden layer unit, h. The first step of this computation is to look at each lower level unit and the bias unit that is connected to the hidden unit. For each of these connections, find the value of the unit and multiply by the weight and sum all the results. The calculations give:

1.0 * 7.1 = 7.10
1.0 *-2.76 = -2.76
0.0 * 7.1 = 0.0
sum = 4.34

In some neural networks we might just leave the activation value of the unit to be 4.34. In this case we would say that we are using the linear activation function, however backprop is at its best when this value is passed to certain types of non-linear functions. The most commonly used non-linear function is:

$$v = 1/(1 + e^{-s})$$

where, s is the sum of the inputs to the neuron and v is the value of the neuron. Thus, with s = 4.34, v = 0.987. This particular function will be called the 'standard sigmoid'. Quite often it is called the logistic function. The general function used to compute the value of a neuron can be called the *activation function* , *squashing function* or *transfer function* . The calculations for the output unit, z are:

1.000 * -4.95 = -4.95
0.000 * -4.95 = 0.00
0.987 * 10.9 = 10.76
1.000 * -3.29 = -3.29
sum = 2.52
v = 0.91

Of course, 0.91 is not quite 1 but for this example it is close enough. When using this particular activation function for a problem where the output is supposed to be a 0 or 1 getting the output to within 0.1 of the target value is a very common standard but other people may want to get to closer than this while others don't want to get even this close. With this particular activation function it is actually somewhat hard to get very close to 1 or 0 because the function only approaches 1 and 0 as the input to the function approaches □ and □. The other values the network computes for the xor function are:

x	y	z
1	0	0.91
0	0	0.08
0	1	1.00
1	1	0.10

The formulas for computing the activation value for a neuron, j can be written more concisely as follows. Let the activation value for neuron j be o_j. Let the activation function be the general function, f. Let the weight between neuron j and neuron i be w_{ij}. Let the net input to neuron j be net_j, then

$$net_j = \sum_{i=1,n} w_{ij}\ o_j$$

where n is the number of units feeding into unit j and $o_j = f(net_j)$