ANN Approach to Analysis the Quality of River Water -A Case Study of Chunnambar River Pondicherry

K.R.Leelavathy[1], V. Nageshwaran[1]
[1]Department of Civil Engineering,
University College of Engineering Tindivanam,
(A Constituent College of Anna University)
Tindivanam, Tamil Nadu, India.

V. Nirmala[2*],
[2]Department of Mathematics,
University College of Engineering Tindivanam,
(A Constituent College of Anna University)
Tindivanam, Tamil Nadu, India.

Abstract **- The water quality management is an important issue of significance in the context of present times. Dissolved oxygen (DO), Biological oxygen demand (BOD), pH, Total Coliforms (Tco) and Temperature (Temp) are some of the important parameters in the characterization of surface water conditions. Our goal is to make a forecast of these parameters in Chunnambar River, Ariyankuppam, Puducherry Region, Southern India. The artificial neural network (ANN) is a powerful computational technique for modeling complex relationship between input and output data. The ANN model was trained using data from the period 2013-2015 and the prediction of water quality was made for the year 2016.The analysis showed good agreement with the water quality index(WQI) which is being traditionally calculated in India. The ANN approach is a practical, simple and useful tool to assess river water quality.**

INTRODUCTION

Artificial Neural Networks (ANNs) are computational modeling tools that have recently emerged and found extensive acceptance in many disciplines for modeling complex real-world problems. ANNs may be defined as structures comprised of densely interconnected adaptive simple processing elements (called artificial neurons or nodes) that are capable of performing massively parallel computations for data processing and knowledge representation(HechtNielsen,1990; Schalkoff, 1997). Although ANNs are drastic abstractions of the biological systems but to make use of what is known about the functionality of the biological networks for solving complex problems. The attractiveness of ANNs comes from the remarkable information processing characteristics of the biological system such as nonlinearity, high parallelism, robustness, fault and failure tolerance, learning, ability to handle imprecise and fuzzy information, and their capability to generalize (Jain et al., 1996). Artificial models possessing such characteristics are desirable because,

i. Nonlinearity allows better fit to the data,
ii. Noise-insensitivity provides accurate prediction in the presence of uncertain data and measurement errors,
iii. High parallelism implies fast processing and hardware failure-tolerance,
iv. Learning and adaptivity allow the system to update its internal structure in response to changing environment, and
v. Generalization enables application of the model to unlearned data.

The main objective of ANN-based computing is to develop mathematical algorithms that will enable ANNs to learn by mimicking information processing and knowledge acquisition in the human brain.

ANN-based models are empirical in nature, however they can provide practically accurate solutions for precisely or imprecisely formulated problems and for phenomena that are only understood through experimental data and field observations. In microbiology, ANNs have been utilized in a variety of applications ranging from modeling, classification, pattern recognition, and multivariate data analysis. Sample applications include,

i. Interpreting pyrolysis mass spectrometry ,GC and HPLC data,
ii. Pattern recognition of DNA, RNA, protein structure, and microscopic images,
iii. Prediction of microbial growth, biomass, and shelf life of food products, and
iv. Identification of microorganisms and molecules.

Sl. No	pollutants	Desirable Limit	Permissible limit in the absence of alternate source
1.	pH	6.5-8.5	No relaxation
2.	BOD (mg/l)	3-5	No relaxation
3.	DO (mg/l)	500	2000
4.	Temperature (°C)	16-27.8	40

Water Quality Index

We are divided into four classes: class I, extraclean fresh ground water resources use forconservation that are not necessary to pass throughwater treatment processes and require only ordinary processes for pathogenic destruction and ecosystem conservation where basic organisms can breed naturally; class II, medium clean fresh ground water resourcesuse for consumption, but are passed through anordinary treatment process before use; class III,fairly clean fresh ground water resources use forconsumption, but requires special water treatmentprocesses before use; and class IV, the sources whichare not within class I to class III and are used for navigation.

Table 1. Indian Water Quality Criteria Classes

Pollutants index	Class I	Class II	Class III	Class IV
pH	6.5-8.5	6.5-8.5	6.5-8.5	6.5<
BOD	2	3	3	>3
DO	6	5	4	4
Temperature				

Classification of ANNs

ANNs may be classified in many different ways according to one or more of their relevant features. Generally, classification of ANNs may be based on (i) the function that the ANN is designed to serve (e.g., pattern association, clustering), (ii) the degree(partial/full) of connectivity of the neurons in the network, (iii) the direction of flow of information within the network (recurrent and non-recurrent), with recurrent networks being dynamic systems in which the state at any given time is dependent on previous states, (iv) the type of learning algorithm, which the represents a set of systematic equations that utilize the outputs obtained from the network along with an arbitrary performance measure to update the internal structure of the ANN, (v) the learning rule (the driving engine of the learning algorithm), and (vi)the degree of learning supervision needed for ANN training. Supervised learning involves training of an ANN with the correct answers (i.e., target outputs) given for every example, and using the deviation (error) of the ANN solution from corresponding target values to determine the required amount by which each weight should be adjusted. Reinforcement learning is supervised, however the ANN is provided with a correct answer itself. Unsupervised learning does not require a correct answer for the training examples, however the network, through exploring the underlying structure in the data and the correlation between the various examples, organizes the examples into clusters (e.g., Kohonen networks). Finally the hybrid learning procedure combines supervised and unsupervised learning.

As examples of classification, Lippmann (1987) classified ANNs with regard to learning (supervised vs unsupervised) and data (binary vs continuous). Simpson (1990) categorized ANNs with respect to learning (supervision) and the flow of data in the network (feedforward vs feedback). Maren (1991) proposed a hierarchal categorization based on structure followed by dynamics, then learning. Jain et al. (1996) present a four-level classified based on the degree of learning supervision, the learning rule, data flow in the ANN, and the learning algorithm.

Development of the ANN model

ANN is a computing tool constructed through many simple interconnected elements called neurons with unique capability of recognizing underlying relationship with input and output events. The structure and operation of ANN are discussed by a numbers of authors (Ranjithan et al., 1993; Rogers and Dowla, 1994; Lippmann, 1987; Hecht-nielsen, 1988; Maren et al., 1990; Burke and lgnizio, 1992; Vemuri, 1988). ANN is arranged into discrete layers, consisting of input, hidden, and output. A collection of neurons arranged conveniently in a dimensional array is called a layer. Each of which includes one or more individual nodes or processing elements. The number of input variables necessary for predicting the desired output variables determines the number of input nodes. The optimum number of hidden nodes and hidden layers are dependent on the complexity of the modeling problem and the objective of the researcher, such as acceptable learning error (Fausett, 1994; Grimes et al., 2003). Greater the number of hidden layers embedded in a dynamic recurrent neural network, larger is the complexity of the model. During training, the ANN is presented with patterns of input and corresponding output pairs. The learning algorithm iteratively adjusts the values of connection weights within the ANN structure. In general, it is more desirable to attain the desired accuracy with a simpler ANN structure (i.e., fewer nodes), as this can reduce training time, improve network generalization and prevent over-fitting (Hagan et al., 1996);

Learning rules

A learning rule defines how exactly the network weights should be adjusted (updated) between successive training cycles (epochs). There are four basic types of rules (Hassoun, 1995; Haykin, 1994). The error-correction learning (ECL) rule is used in supervised learning in which the arithmetic difference (error) between the ANN solution at any stage (cycle) during training and the corresponding correct answer is used to modify the connection weights so as to gradually reduce the overall network error. The Boltzmann learning (BL) rule is a stochastic rule derived from thermodynamic principles and information theory (Anderson and Rosenfeld, 1988). It is similar to ECL, however each neuron generates an output (or state) based on a Boltzmann statistical distribution (Jain et al., 1996), which renders learning extremely slower. The Hebbian learning (HL) rule (Hebb, 1949), developed based on neurobiological experiments, is the oldest learning rule, which postulates that “if neurons on the both sides of a synapse are activated synchronously and repeatedly, the

synapse's strength is selectively increased." Therefore, unlike ECL and BL rules, learning is done locally by adjusting the synapse weight based on the activities of neurons. In the competitive learning (CL) rule, all neurons are forced to compete among themselves such that only one neuron will be activated in a given iteration with all the weights attached to it adjusted (Jain et al., 1996). The CL rule is speculated to exist in many biological systems (Haykin, 1994).

Popular ANNs

A vast number of networks, new or modifications of existing ones, are being constantly developed. Simpson (1990) listed 26 different types of ANNs, and Maren (1991) listed 48. Pham (1994) estimated that over 50 different ANN types exist. Some applications may be solved using different ANN types, where as others may only be solved via a specific ANN type. Some networks are more proficient in solving perceptual problems, while others are more suitable for data modeling and function approximation. A brief discussion of the most frequently used ANNs, presented in the order of their discovery, is given below.

Hopfield networks

This network is a symmetric fully connected two layer recurrent network that acts as a nonlinear associative memory and is especially efficient I solving optimization problems (Hopfield, 1984; Hopfield and Tank, 1986). The network is suited to only bipolar or binary inputs and it implements an energy function. Learning is done by setting each weight connecting two neurons to the product of the inputs of these two neurons (Van Rooij et al., 1996). When presented with an incomplete or noisy pattern, the network responds by retrieving an internally stored pattern that most closely resembles the presented pattern.

Adaptive resonance theory (ART) networks

These are trained by unsupervised learning where the network adapts to the information environment without intervention. The ART network consists of two fully interconnected layers, a layer that receives the inputs and a layer consisting of output neurons. The feedforward weights are used to select the winning output neuron (cluster) and serve as the long-term memory for the networks. The feedback weights are the vigilance weights that are used to test the vigilance and serve as the short-term memory for the network (Van Rooij et al., 1996). An ART network stores a set of patterns in such a way that when the network is presented with a new patterns in such a way that when the networks is presented with a new pattern it will either match it to a previously stored pattern, or store it as a new pattern if it is sufficiently dissimilar to the closest pattern (Carpenter and Grossberg, 1987, 1988). Like Hopfield nets, ART networks can be used for pattern recognition, completion, and classification.

Kohonen networks

These networks, also called self-organizing feature maps, are two-layer networks that transform n-dimensional input patterns into lower-ordered data where similar patterns project onto points in close proximity to one another (Kohonen, 1989). Kohonen networks are trained in an unsupervised manner to form clusters within data (i.e., data grouping). In addition to pattern recognition and classification, Kohonen maps are used for data compression, in which high-dimensional data are mapped into a fewer dimensions space while preserving their content (Zupan and Gasteiger, 1991).

Back propagation networks

These networks are the most widely used type of networks and are considered the workhorse of ANNs (Rumelhart et al., 1986). A backpropagation (BP) network is an MLP consisting of

i. An input layer with nodes representing input variables to the problem,
ii. An output layer with nodes representing the dependent variables (i.e., what is being modeled),
iii. One or more hidden layers containing nodes to help capture the nonlinearity in the data.

Using supervised learning (with the ECL rule), these networks can learn the mapping from one data space to another using examples. The term backpropagation refers to the way the error computed at the output side is propagated backward from the output layer. In BPANNs, the data are fed forward into the network without feedback (i.e., all links are unidirectional and there are n same layer neuron-to-neuron connections). The networks are so versatile and can be used for the data modeling, classification, forecasting, control, data and image compression, and pattern recognition (Hassoun, 1995).

Radial basis function (RBF) networks

These networks are a special case of a multilayer feedforward error-backpropagation network with three layers (Schalkoff, 1997).They can be trained by a variety of learning algorithms including a two-step hybrid learning (Haykin, 1994). The hidden layer is used to cluster the inputs of the network (the nodes in this layer are called cluster centers). Un like the sigmoid transfer function in BPANNs, these networks employ a radial basis function such as a Gaussian kernel (Haykin, 1994). The RBF is centered at the point specified by the weight vector associated with the unit. Both the positions and widths of these Gaussian functions must be learnt from the training patterns. Each output unit implements alinear combination of these RBFs. The choice between the RBF networks and the BPANNs is problem dependent (Pal and Srimani, 1996). RBF networks train faster than BP but are as versatile and comparatively slower for use (Attoth-Okine etal., 1996).

The decision as to which network works better for a give problem depends strictly on the problem logistics. For example ,a clustering problem requires a Kohonen network, a mapping problem may be modeled using a

variety of ANNs such as BP and RBF networks, and some optimization problems may only be solved using Hopfield networks. Other factors governing ANN selection are the input type (i.e., whether it is Boolean, continuous, or a mixture), and the execution speed of the network once trained and implemented in serial hardware. Other issues for ANN selection are discussed by Hudson and postma (1995).

Backpropagation ANNs

To extend the understanding of ANNs from the level of identifying what these systems are to how to design them, it is imperative to become familiar with ANN computation and design. For this objective, the BPANNs are discussed in more detail, for their popularity, and their flexibility and adaptability in modeling a wide spectrum of problems in many application areas.

The feedforward error-backpropagation learning algorithm is the most famous procedure for training ANNs. BP is based on searching an error surface (error as a function of ANN weights) using gradient descent for point(s) with minimum error. Each iteration in BP constitutes two sweeps: forward activation to produce a solution, and a backward propagation of the computed error to modify the weights. In an initialized ANN (i.e., an ANN with assumed initial weights), the forward sweep involves presenting the network with one training example. This starts at the input layer where each input node transmits the value received forward to each hidden layer. The collective effect on each of the hidden nodes is summed up by performing the dot product of all values of input nodes and their corresponding interconnection weights, as described in Eq.(1). Once the net effect at one hidden node is determined, the activation at the node is calculated using a transfer function (e.g., sigmoidal function) to yield an output between 0 and +1 or -1 and +1. The amount of activation obtained represents the new signal that is to be transferred forward to the subsequent layer (e.g., either hidden or output layer). The same procedure of calculating the net effect is repeated for each hidden node and for all hidden layers. The net effects calculated at the output nodes is consequently transformed into activations using a transfer function. The activations just calculated at the output nodes represents the ANN solution of the fed example, which may deviate considerably from the target solution due to the arbitrary selected interconnection weights. In the ANN and target outputs is used to adjust the interconnection weights, starting from the output layer, through all hidden layers, to the input layers, as will be described in the following section. The forward and backward sweeps are performed repeatedly until the ANN solution agrees with the target value within a prespecified tolerance. The BP learning algorithm provides the needed weight adjustments in the backward sweep.

BP Algorithm

Because of its importance and simplicity, the BP algorithm will be presented here in its final form. Complete derivation of the algorithm can be found elsewhere (e.g., Zupan and Gasteiger, 1993; Haykin, 1994) and a clear systematic derivation is given by Basheer (1998) and Najjar et al. (1997). In order to be able to run the algorithm, it is essential to define the interlayer as the gap between two successive layers that encloses the connection weights and contains only the neurons of the upper layer. Consider an MLP network with L interlayers. For interlayer l€{1,2,….,L} there are N_l nodes and $N_l X N_{l-1}$ connection links with weights W € $R^{N_l X N_{l-1}}$, where N_l and N_{l-1} are the numbers of nodes (including thresholds) in interlayers l and $l-1$, respectively. A connection weight is denoted by W_{ji}^l if it resides in interlayer l and connects node j of interlayers $l$with node I of lower (predicting) interlayer $l-1$(node i is the source node and node j is the destination node). In any interlayer l, a typical neuron j integrates the signals, x_j, impinging onto it, and produces a net effect, j , according to linear neuron dynamics:

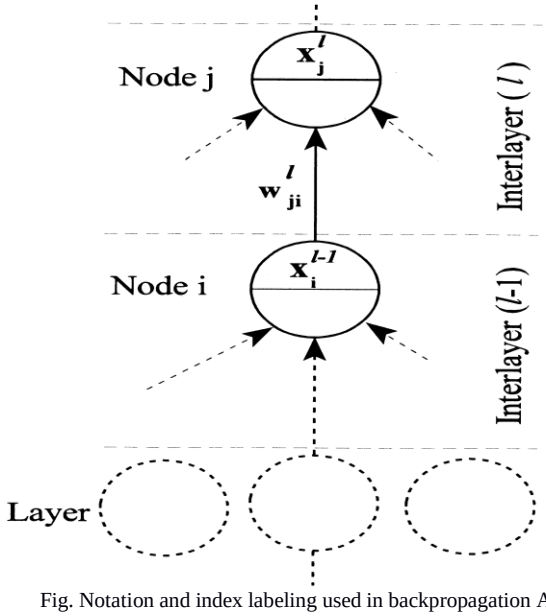

Fig. Notation and index labeling used in backpropagation ANNs.

$$\xi_j^l = \sum_{i=1}^{N_{l-1}} W_{ji}^l x_i^{l-1}. \quad (1)$$

The corresponding activation, x_j^l, of the neuron is determined using a transfer function, σ,that converts the total signal into a real number from a bounded interval:

$$x_j^l = \sigma(\xi_j^l) = \sigma\left(\sum_{i=1}^{N_{l-1}} W_{ji}^l x_i^{l-1}\right). \quad (2)$$

One popular function used in BP is the basic continuous sigmoid:

$$\sigma(\xi) = \frac{1}{1+e^{-\xi}} \quad (3)$$

Where $-\infty < \xi < \infty$ and $0.0 < \sigma < 1.0$. Eqs.(1)-(3) Are used for all nodes the activation is simply the raw input. In any interlayer, an arbitrary weight W_{ji}^l at iteration (t) will be updated from its previous state (t-1) value according to

$$W_{ji}^l(t) = W_{ji}^l(t\text{-}1) + \Delta W_{ji}^l(t), \quad (4)$$

Where ΔW_{ji}^{l} is the (+ / -) incremental change in the weight. The weight change is determined via the modified delta rule (Zupan and Gasteiger, 1993). Which can be written as

$$\Delta W_{ji}^{l} = \eta \delta_j^l x_i^{l-1} + \mu \Delta W_{ji}^{l(previous)}, \quad (5)$$

Where η is the learning rate controlling the update step size, μ is the momentum coefficient, and x_i^{l-1} is the input from the l -1th interlayer. The first part of the right-hand side of Eq. (5) is the original delta rule. The added momentum term helps direct the search on the error hyperspace to the global minimum by allowing a portion of the previous updating (magnitude and direction) to be added to the current updating step. Note that Eq. (5) can also be applied to any neuron threshold (bias) which can be assumed as a link, with weight equal to the threshold value, for an imaginary neuron whose activation is fixed at 1.0. The weight change can also be determined using a gradient descent written in generalized form for an interlayer :

$$\Delta W_{ji}^{l} = -k \left(\frac{\partial \varepsilon^l}{\partial w_{ji}^l}\right). \quad (6)$$

Therefore, in order to determine the incremental changes for the lth interlayer, the main task is to quantify the error gradient $\left(\frac{\partial \varepsilon^l}{\partial w_{ji}^l}\right)$. Using Eq. (5) and (6), the required weight change can be derived with different expressions depending on whether the considered neuron is in the output layer, then $l = L$ in Eq. (5), with δ_j^L calculated from ,

$$\delta_j^L = (x_j^L - y_j)\, x_j^L(1- x_j^L) \quad (7)$$

If the neuron is in a hidden layer, the weight change is also calculated using eq. (6) with δ_j^L determined from,

$$\delta_j^l = x_j^l(1- x_j^l)\left(\sum_{k=1}^{r} \delta_k^{l+1} w_{kj}^{l+1}\right) \quad (8)$$

Where δ_k^{l+1} is calculated for a given non-output layer (l) beginning with a layer one layer one level up ($l + 1$and moving down layer by layer. That is, for the last (uppermost) hidden layer in a network, δ_j^l is determined via $\delta_k^{l+1}$of the output layer calculated using eqn.(8). The above delta equations (Eqs. (7) and (8) are based on the sigmoid transfer function given in Eq. (3). For a different function, the terms $x_j^L(1- x_j^L)$ and $x_j^l(1- x_j^l)$ in Eqs. (7) and (8), respectively, should be replaced with the relevant first derivative of the used function. This technique of distributing backward the errors starting from the output layer down through the hidden layer gave the method the name backpropagation of the errors starting from the output layer down through the hidden layer gave the method the name backpropagation of error with the modified delta rule (Rumelhard et al., 1986). The standard BP have been modified in several ways to achieve a better search and accelerate and stabilize the training process (Looney, 1996; Masters, 1994).

SIMULATION RESULTS AND DISCUSSION

The proposed system model for WQI prediction was validated using simulation studies. The studies were carried out by using MATLAB simulation environment. For the validation of the model, the water samples collected from the Chunnambar River were used. According to Table 2, the designed model predicts unfit category for Chunnambar River, for all the months from March 2013 to February 2016. The predictions were close to the Indian standard results. It is quite evident that the rivers are receiving huge amount of organic waste, when it passes through an urban setting. This has been caused due to even increasing population. The solution of this problem lies in the treatment of sewage and disposal of fully or partially treated sewage waste. Though, the processing is quite expensive, the treatment process will have to be followed in order to save river.

Table 2. Simulation results for prediction of WQI in Chunnambar River as per Indian Water Quality Criteria.

Sl. No.	Month	pH	BOD (mg/l)	DO (mg/l)	Temp. (°C)	WQI	Prediction of Result by FIS
1.	March'13	7.28	150.24	2.40	29.3	Unfit	20.0000
2.	April'13	7.35	140.45	2.58	29.0	Unfit	20.0000
3.	May'13	7.41	121.32	3.32	29.4	Unfit	20.0000
4.	June'13	7.40	123.48	3.14	29.2	Unfit	20.0000
5.	July'13	7.37	122.51	3.26	28.2	Unfit	20.0000
6.	August' 13	7.41	135.72	3.65	28.0	Unfit	20.0000
7.	September'13	7.42	110.75	4.00	27.1	Unfit	20.0000
8.	October'13	7.40	119.42	3.79	27.5	Unfit	20.0000
9.	November'13	7.43	88.91	4.97	26.4	Unfit	20.0000
10.	December'13	7.51	59.82	5.57	25.3	Unfit	20.0000
11.	January'14	7.49	71.63	4.91	25.7	Unfit	20.0000
12.	February'14	7.48	90.10	4.28	26.8	Unfit	20.0000
13.	March'14	7.43	149.50	2.89	29.4	Unfit	20.0000
14.	April'14	7.47	141.65	2.97	29.2	Unfit	20.0000
15.	May'14	7.52	120.37	3.53	29.5	Unfit	20.0000
16.	June'14	7.49	123.60	3.34	29	Unfit	20.0000
17.	July'14	7.35	121.11	3.18	28.3	Unfit	20.0000
18.	August' 14	7.29	135.38	3.55	28.1	Unfit	20.0000
19.	September'14	7.45	109.55	3.98	27.2	Unfit	20.0000
20.	October'14	7.38	120.42	3.56	27.4	Unfit	20.0000
21.	November'14	7.46	89.02	5.00	26.7	Unfit	20.0000
22.	December'14	7.40	60.12	5.66	25.3	Unfit	20.0000

23.	January'15	7.32	71.77	4.83	25.8	Unfit	20.0000
24.	February'15	7.48	90.64	4.16	26.4	Unfit	20.0000
25.	March'15	7.37	151.21	2.56	29.3	Unfit	20.0000
26.	April'15	7.41	140.34	2.74	29.1	Unfit	20.0000
27.	May'15	7.52	122.38	3.52	29.2	Unfit	20.0000
28.	June'15	7.46	124.43	3.05	29.4	Unfit	20.0000
29.	July'15	7.35	123.68	3.32	28.4	Unfit	20.0000
30.	August' 15	7.42	135.20	3.45	28.2	Unfit	20.0000
31.	September'15	7.39	119.75	3.98	27.3	Unfit	20.0000
32.	October'15	7.48	121.62	3.57	27.6	Unfit	20.0000
33.	November'15	7.51	89.91	4.97	26.2	Unfit	20.0000
34.	December'15	7.27	61.83	5.67	25.1	Unfit	20.0000
35.	January'16	7.38	72.64	4.81	25.7	Unfit	20.0000
36.	February'16	7.45	90.11	4.25	26.5	Unfit	20.0000

CONCLUSION

The FIS has been used to predict water quality for Chunnambar River, Ariyankuppam, Puducherry. This proposed model gives very close results as compared with Indian surface water quality criteria. The applicability and usefulness of the proposed methodology was revealed by a case study.

ACKNOWLEDGMENTS

The authors wish to thank R.Kowsalya, E.Lingeshwari, P.Ranjitha, J.Subashini, R.Vijayalakshmi the students of B.E. Civil Engineering, University College of Engineering Tindivanam, for their support and contribution towards collecting and analyzing the water samples from the Chunnambar River, Puducherry.

REFERENCES

[1] Hecht-Nielsen, R., 1990. Neurocomputing. Addidon-Wesley, Reading, MA.
[2] Schalkoff, R.J., 1997. Artificial Neural Networks. McGraw-Hill, New York.
[3] Jain, A.K., Mao, J., Mohiuddin, K.M., 1996. Artificial neural networks: a tutorial. Comput. IEEE March, 31-44.
[4] Kohonen, T., 1989. Self-organization and Associative Memory, 3rd Edition. Springer, New York.
[5] Lippmann, R.P., 1987. An introduction to computing with neural nets. IEEE ASSP Mag. 4, 4-22.
[6] Simpson, P.K., 1990. Aritificial Neural Systems: Foundations, Paradigms, Applications, and Implementations. Pergamon Press, New York.
[7] Maren, A.J., 1991. A logical topology of neural networks. In: Proceedings of the Second Workshop on Neural Networks, WNN-AIND 91.
[8] Ranjithan, S., Eheart, J.W., Garrett, J.H., 1993. Neural network based screening for groundwater reclamation under uncertainty. Water Resour. Res. 29 (3), 563–574.
[9] Rogers, L.L., Dowla, F.U., 1994. Optimization of groundwater remediation using artificial neural network with parallel solute transport modeling. Water Resour. Res. 30 (2), 457–481.
[10] Burke, L. I., Ignizio, J.P., 1992. Neural networks and operations research: an overview. Comput. Oper. Res. 19 (3/4), 179–189.
[11] Vemuri, V., 1988. Artificial neural networks: an introduction. In: Vemuri, V. (Ed.), Artificial Neural Networks: Theoretical Concepts. Comput. Soc. Press, Institute of Electrical and Electronics Engineers, NY, pp. 1-12.
[12] Fausett, L., 1994. Fundamentals of Neural Networks. Prentice-Hall, Englewood Cliffs, NJ.
[13] Grimes, D.I.F., Coppola, E., Verdecchia, M., Visconti, G., 2003. A neural network approach to real-time rainfall estimation for Africa using Satellite data. J. Hydromet. 4 (6), 1119–1133.
[14] Hagan, M.T., Demuth, H.B., Beale, M.H., 1996. Neural Network Des.. PWS, Boston, MA.
[15] Hassoun, M.H., 1995. Fundamentals of Artificial Neural Net works. MIT Press, Cambridge, MA.
[16] Haykin, S., 1994. Neural Networks: A Comprehensive Foundation. Macmillan, New York.
[17] Hebb, D.O., 1949. The Organization of Behavior. Wiley, New York.
[18] Pham, D.T., 1994. Neural networks in engineering. In: Rzevski, G. et al. (Eds.), Applications of Artificial Intelligence in Engineering IX, AIENG/ 94, Proceedings of the 9th InternationalConference. Computational Mechanics Publications, Southampton, pp. 3–36.
[19] Hopfield, J.J., 1984. Neurons with graded response have collective computational properties like those of two-state neurons. ProcNatl. Acad. Sci. 81, 3088–3092.
[20] Hopfield, J.J., Tank, D.W., 1986. Computing with neural circuits: a model. Science 233, 625–633.
[21] van Rooij, A., Jain, L., Johnson, R., 1996. Neural NetworkTraining Using Genetic Algorithms. World Scientific, Singa-pore
[22] Carpenter, G.A., Grossberg, S., 1988. The ART of adaptive pattern recognition by a self-organizing neural network. Computer March, 77–88.
[23] Carpenter, G.A., Grossberg, S., 1987. ART2: self-organization of stable category recognition codes for analog input patterns. Appl. Opt. 26, 4914–4930.
[24] Zupan, J., Gasteiger, J., 1993. Neural Networks For Chemists: An Introduction. VCH, New York
[25] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning internal representation by error propagation. In: Rumelhart, D.E., McClleland, J.L. (Eds.). Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. 1. MIT Press, Cambridge, MA, Chapter 8.
[26] Pal, S.K., Srimani, P.K., 1996. Neurocomputing: motivation, models, and hybridization. Computer March, 24–28.
[27] Attoh-Okine, N., Basheer, I., Chen, D.-H., 1999. Use of artificial neural networks in geomechanical and pavement systems. Trans. Res. Board, Circular E-C012.
[28] Hudson, P., Postma, E., 1995. Choosing and using a neural net. In: Braspenning, P.J. et al. (Eds.), Artificial Neural Networks, An Introduction to ANN Theory and Practice. Lecture Notes in Computer Science. Springer, NY, pp. 273–287.
[29] Basheer, I.A., 1998. Neuromechanistic-based modeling and simu lation of constitutive behavior of fine-grained soils. PhD Dissertation, Kansas State University, 435 pp.
[30] Najjar, Y., Basheer, I., Hajmeer, M., 1997. Computational neural networks for predictive microbiology. I. Methodology. Int. J. Food Microbiol. 34, 27–49.
[31] Looney, C.G., 1996. Advances in feedforward neural networks: demystifying knowledge acquiring black boxes. IEEE Trans. Knowledge Data Eng. 8 (2), 211–226.
[32] Masters, T., 1994. Practical Neural Network Recipes in C11. Academic Press, Boston, MA.