HYDROLOGICAL DATA PROCESSING AND SOFT COMPUTING TECHNIQUES IN GW STUDIES

Anil Kumar Lohani
Scientist G
Surface Water Hydrology Division
National Institute of Hydrology, Roorkee

Introduction

The quality control and long-term archiving of groundwater quantity data represent a central function of Central/State hydrometric agencies. This should take a user-focused approach to improving the information content of datasets, placing strong emphasis on maximising the final utility of data e.g. through efforts to improve completeness and fitness-for-purpose of Centrally/State archived data. It includes the process from data entry through primary and secondary validation, to analysis of groundwater level data. During all levels of validation, staff should be able to consult station metadata records detailing the history of the site and its hydrometric performance, along with hydrogeological and climate maps and previous quality control logs. Numerical and visual tools available at different phases of the data validation process, such as versatile groundwater level plotting and manipulation software to enable comparisons between different near-neighbour or analogue observation wells and assessment of time series statistics greatly facilitate validation.

For the aanalysis of hydro-meteorological and ground water data reserarchers have demonstrated use of soft computing techniques e.g. ANN and Fuzzy Logic. The research in Artificial Neural Networks (ANNs) started with attempts to model the biophysiology of the brain, creating models which would be capable of mimicking human thought processes on a computational or even hardware level. Humans are able to do complex tasks like perception, pattern recognition, or reasoning much more efficiently than state-of-the-art computers. They are also able to learn from examples and human neural systems are to some extent fault tolerant.

Data processing

The processing of groundwater level data starts with preliminary checking in the field. When making groundwater level measurements, the observer should always note any occurrences which may influence the groundwater level as observed by the instruments. These may include: damage to the equipment for a specified reason. The observer should also note any maintenance activities carried out at the monitoring site (e.g. change batteries, clean sensor, etc). The observer should double-check that that any manual reading is taken correctly, and transcribed correctly (e.g. decimal point in right place). If the reading is later transferred to another document (e.g. hand copied or typed in, or abstracted from a chart), the observer should always check that this has been done correctly. An experienced and suitably qualified observer should compare measurements with equivalent ones from earlier that day or from the day before, if available, as an additional form of checking. However, the observer should not, under any circumstances, retrospectively alter earlier readings or adjust current readings, but should simply add an appropriate comment.

Validation ensures that the data stored are as complete and of the highest quality as possible by: identifying errors and sources of errors to mitigate them occurring again, correcting errors where possible, and assessing the reliability of data. Data validation is split into two principal stages: primary and secondary, with an optional tertiary stage. Validation is very much a two-way process, where each step feeds back to the previous step any comments or queries relating to the data provided. The data processing steps comprise:

1. Receipt of data according to prescribed target dates. Rapid and reliable transfer of data is essential, using the optimal method based on factors such as volume, frequency, speed of transfer/transmission and cost. Maintenance of a strict time schedule is important because it gives timely feedback to monitoring sites, it encourages regular exchanges between field staff, Sub-Divisional offices, State and Central agencies, it creates continuity of processing activities at different offices, and it ensures timely availability of final (approved) data for use in policy and decision-making.

2. Entry of data to computer, is primarily done at a Sub-Divisional office level where staff are in close contact to field staff who have made the observations and/or collected the digital data. Historical data, previously only available in hardcopy form, may also be entered this way. Each Central/State agency should have a programme of historical data entry.

3. Primary data validation which should be carried out in State DPCs for State data and CGWB local offices for CGWB data, as soon as possible after the observations are made or data downloaded from loggers, using e-GEMS. This ensures that any obvious problems (e.g. indicating an instrument malfunction, observer error, etc) are spotted at the earliest opportunity and resolved. Other problems may not become apparent until more data have been collected, and data can be viewed in a longer temporal context during secondary validation.

4. Secondary data validation which should be carried out in State DPCs for State data and CGWB local offices for CGWB data, to take advantage of the information available from a large area by focusing on comparisons with the same variable at other good quality, nearby monitoring sites (analogue stations) which are expected to exhibit similar hydrogeological behaviours, uses e-GEMS. States should have access to CGWB data during secondary validation, and may receive support from CGWB in this activity.

5. Tertiary data validation which focuses on advanced techniques for the analysis and validation of spatial and temporal data, using tools like statistical analysis and spatial overlays. This stage of validation is time-consuming and is applied selectively.

Measurement errors for groundwater level data

Manual depth to groundwater level measurement errors

- Observer reads water depth incorrectly
- Observer enters water depth incorrectly in the field sheet (e.g. misplacement of decimal point in the range 0.01 to 0.10, writing 4.9 m instead of 4.09 m)
- Observer enters groundwater level to the wrong day or time
- Observer fabricates readings, indicated by sudden changes in levels or extended periods of uniform mathematical sequences of observations
- Observer uses incorrect measurement point
- Observer enters depth of well for dry well
- Depth of well is greater than length of measuring tape

DWLR measurement errors

- Failure of electronics due to lightning strike etc. (though lightning protection usually provided)
- Incorrect set up of measurement parameters by the observer or field supervisor

Primary validation of groundwater level data focuses on validation within a single data series by making comparisons between individual observations and physical limits, and between two measurements of groundwater level at a single station (e.g. a DWLR groundwater level and a manually-read check of depth to groundwater level).

Secondary validation focuses on further investigation of data from the well, including comparison between groundwater level and incident rainfall (initial), and comparisons with neighbouring wells to identify suspect values (intermediate). Data processing staff should continue to be aware of field practice and instrumentation and the associated errors which can arise in data.

Artificial Neural Networks

The most common ANNs consist of several layers of simple processing elements called neurons, interconnections among them and weights assigned to these interconnections. The information relevant to the input–output mapping of the net is stored in the weights. A neural network is not programmed like a conventional computer program, but is presented with examples of the patterns, observations and concepts, or any type of data which it is supposed to learn. Through the process of learning (also called training) the neural network organizes itself to develop an internal set of features that it uses to classify information or data. Due to its massively parallel processing architecture the ANN is capable of efficiently handling complex computations, thus making it the most preferred technique today for high speed processing of huge data. Other advantages include:

1. ***Adaptive learning:*** An ability to learn how to do tasks based on the data given for training or initial experience.
2. ***Self-Organisation:*** An ANN can create its own organisation or representation of the information it receives during learning time.
3. ***Real Time Operation:*** ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. ***Fault Tolerance via Redundant Information Coding:*** Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

These properties make ANN suitable candidates for various engineering applications such as pattern recognition, classification, function approximation, system identification, hydrological modeling etc.

This lecture describes the concept of artificial neuron, ANN structure, back propagation algorithm, and ANN applications in hydrology.

Biological Neuron

It is claimed that the human central nervous system is comprised of about 1,3x1010 neurons and that about 1x1010 of them takes place in the brain. At any time, some of these neurons are firing and the power dissipation due this electrical activity is estimated to be in the order of 10 watts. A neuron has a roughly spherical cell body called soma (Figure 1). The signals generated in soma are transmitted to other neurons through an extension on the cell body called ***axon*** or ***nerve fibres.*** Another kind of extensions around the cell body like bushy tree is the ***dendrites***, which are responsible from receiving the incoming signals generated by other neurons.

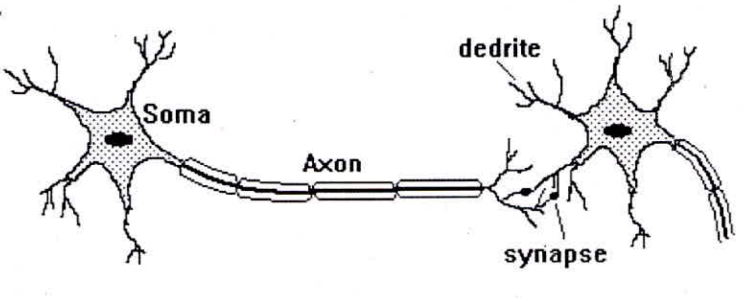

Figure 1: Typical Neuron

As it is mentioned in the previous section, the transmission of a signal from one neuron to another through synapses is a complex chemical process in which specific transmitter substances are released from the sending side of the junction. The effect is to raise or lower the electrical potential inside the body of the receiving cell. If this graded potential reaches a threshold, the neuron fires. It is this characteristic that the artificial neuron model proposed by McCulloch and Pitts, (McCulloch and Pitts 1943) attempt to reproduce.

Research into *models* of the human brain already started in the 19th century (James, 1890). It took until 1943 before McCulloch and Pitts (1943) formulated the first ideas in a mathematical model called the McCulloch-Pitts neuron. In 1957, a first multilayer neural network model called the perceptron was proposed. However, significant progress in neural network research was only possible after the introduction of the backpropagation method (Rumelhart, et al., 1986), which can train multi-layered networks.

Artificial Neuron

Mathematical models of biological neurons (called artificial neurons) mimic the functionality of biological neurons at various levels of detail. A typical model is basically a static function with several inputs (representing the dendrites) and one output (the axon). Each input is associated with a weight factor (synaptic strength). The weighted inputs are added up and passed through a nonlinear function, which is called the *activation function* (ASCE, 2000a; APPENDIX-I). The value of this function is the output of the neuron (Figure 2).

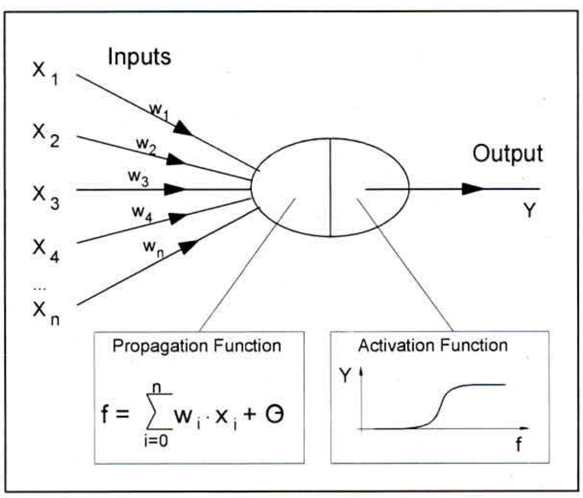

Figure 2: Processing Element of ANN

Neural Network Architecture

A typical ANN model consists of number of layers and nodes that are organised to a particular structure. There are various ways to classify a neural network. Neurons are usually arranged in several *layers* and this arrangement is referred to as the *architecture* of a neural net. Networks with several layers are called *multi-layer* networks, as opposed to *single-layer* networks that only have one layer. The classification of neural networks is done by the number of layers, connection between the nodes of the layers, the direction of information flow, the non linear equation used to get the output from the nodes, and the method of determining the weights between the nodes of different layers. Within and among the layers, neurons can be interconnected in two basic ways: (1) *Feedforward networks* in which neurons are arranged in several layers. Information flows only in one direction, from the input layer to the output layer, and (2) *Recurrent networks* in which neurons are arranged in one or more layers and feedback is introduced either internally in the neurons, to other neurons in the same layer or to neurons in preceding layers. The commonly used neural network is three-layered feed forward network due to its general applicability to a variety of different problems and is presented in Figure 3

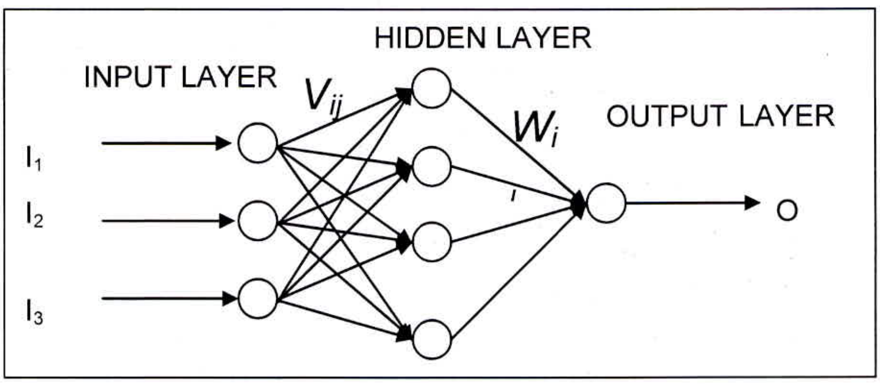

Figure 3: A Typical Three-Layer Feed Forward ANN (ASCE, 2000a)

Learning

The learning process in biological neural networks is based on the change of the interconnection strength among neurons. Synaptic connections among neurons that simultaneously exhibit high activity are strengthened. In artificial neural networks, various concepts are used. A mathematical approximation of biological learning, called Hebbian learning is used, for instance, in the Hopfield network. Multi-layer nets, however, typically use some kind of optimization strategy whose aim is to minimize the difference between the desired and actual behavior (output) of the net. Two different learning methods can be recognized: supervised and unsupervised learning:

Supervised learning: the network is supplied with both the input values and the correct output values, and the weight adjustments performed by the network are based upon the error of the computed output.

Unsupervised learning: the network is only provided with the input values, and the weight adjustments are based only on the input values and the current network output. Unsupervised learning methods are quite similar to clustering approaches.

Multi-Layer Neural Network

A multi-layer neural network (MNN) has one input layer, one output layer and a number of hidden layers between them. In a MNN, two computational phases are distinguished:

1. *Feedforward computation.* From the network inputs (*xi, i = 1, . . . , n)*, the outputs of the first hidden layer are first computed. Then using these values as inputs to the second hidden layer, the outputs of this layer are computed, etc. Finally, the output of the network is obtained.
2. *Weight adaptation.* The output of the network is compared to the desired output. The difference of these two values called the error, is then used to adjust the weights first in the output layer, then in the layer before, etc., in order to decrease the error. This backward computation is called error backpropagation. The error backpropagation algorithm was proposed by and Rumelhart, et al. (1986) and it is briefly presented in the following section.

Feed forward Computation

In a multi layer neural network with one hidden layer, step wise the feed forward computation proceeds as:

I. Forward Pass

Computations at Input Layer

Considering linear activation function, the output of the input layer is input of input layer:

$$O_l = I_l \qquad (1)$$

where, O_l is the l^{th} output of the input layer and I_l is the l^{th} input of the input layer.

Computations at Hidden Layer

The input to the hidden neuron is the weighted sum of the outputs of the input neurons:

$$I_{hp} = u_{1p}O_1 + u_{2p}O_2 + ...+ u_{lp}O_l \qquad (2)$$

for $p = 1,2,3,.....m$

where, I_{hp} is the input to the p^{th} hidden neuron, u_{lp} is the weight of the arc between l^{th} input neuron to p^{th} hidden neuron and m is the number of nodes in the hidden layer.

Now considering the sigmoidal function the output of the p^{th} hidden neuron is given by:

$$O_{hp} = \frac{1}{(1+e^{-\lambda(I_{hp}-\theta_{hp})})} \qquad (3)$$

where O_{hp} is the output of the p^{th} hidden neuron, I_{hp} is the input of the p^{th} hidden neuron, θ_{hp} is the threshold of the p^{th} neuron and λ is known as sigmoidal gain. A non-zero threshold neuron is computationally equivalent to an input that is always held at -1 and the non-zero threshold becomes the connecting weight values.

Computations at Output Layer

The input to the output neurons is the weighted sum of the outputs of the hidden neurons:

$$I_{Oq} = w_{1q}O_{h1} + w_{2q}O_{h2} + \ldots\ldots + w_{mq}O_{hm} \quad (4)$$

$$\text{for } q = 1,2,3,\ldots.n$$

where, I_{Oq} is the input to the q^{th} output neuron, w_{mq} is the weight of the arc between m^{th} hidden neuron to q^{th} output neuron.

Considering sigmoidal function, the output of the q^{th} output neuron is given by:

$$O_{Oq} = \frac{1}{(1+e^{-\lambda(I_{Oq}-\theta_{Oq})})} \quad (5)$$

where, O_{Oq} is the output of the q^{th} output neuron, λ is known as sigmoidal gain, θ_{Oq} is the threshold of the q^{th} neuron. This threshold may also be tackled again considering extra 0^{th} neuron in the hidden layer with output of -1 and the threshold value θ_{Oq} becomes the connecting weight value.

Computation of Error

The error in output for the r^{th} output neuron is given by:

$$\xi^l = \frac{1}{2}\sum_{r=1}^{n}(T_{Or} - O_{or})^2 \quad (6)$$

where O_{Or} is the computed output from the r^{th} neuron and T_{Or} is the target output.

Equation (4.19) gives the error function in one training pattern. Using the same technique for all the training patterns the error function become

$$\xi = \sum_{j=1}^{N}\xi^j \quad (7)$$

where, N is the number of input-output data sets.

Training of Neural Network

Training is the adaptation of weights in a multi-layer network such that the error between the desired output and the network output is minimized.

II. Backword Pass

For k^{th} output neuron, E_k is given by

$$\xi_k = \frac{1}{2}(T_k - O_{ok})^2 \tag{8}$$

where, T_k is the target output of the k^{th} output neuron and O_{ok} is the computed output of the k^{th} output neuron. The output of the k^{th} output neuron is given by

$$O_{Ok} = \frac{1}{(1 + e^{-\lambda(I_{Ok} - \theta_{Ok})})} \tag{9}$$

The change of weight for weight adjustment of synapses connecting hidden neurons and output neurons is expressed as:

$$\Delta w_{ik} = -\eta \frac{\partial \xi_k}{\partial w_{ik}} = -\eta \cdot O_{hi} \cdot d_k \tag{10}$$

where, $d_k = \lambda \cdot (T_K - O_{Ok}) \cdot O_k \cdot (1 - O_{Ok})$ and η is learning rate constant

Learning rate coefficient determines the size of the weight adjustment made at each iteration and hence influences the rate of convergence. Poor choice of the learning coefficient can result in a failure in convergence. For a too large learning rate coefficient the search path will oscillate and jump past the minimum. For a very small learning rate coefficient the descent will progress in a small steps and thus significantly increase the time of convergence.

Therefore, change of weight for weight adjustment of synapses connecting input neurons and hidden neurons is expressed as:

$$\Delta u_{ij} = -\eta \frac{\partial \xi_k}{\partial u_{ij}} = -\eta [\{-w_{ik} d_k\} \cdot \{\lambda(O_{hi})(1 - O_{hi})\} \cdot \{I_{ij}\}] \tag{11}$$

The performance of the backpropagation algorithm depends on the initial setting of the weights, learning rate, output function of the units (sigmoidal, hyperbolic tangent etc.) and the presentation of training data. The initial weights should be randomized and uniformly distributed in a small range of values. Learning rate is important for the speed of convergence. Small values of learning parameter may result in smooth trajectory in the weight space but takes long time to converge. On the other hand large values may increase the learning speed but result in large random fluctuations in the weight space. It is desirable to adjust the weights in such a way that all the units learn nearly at the same rate. The training data should be selected so that it represents all data and the process adequately. The major limitation of the backpropagation algorithm is its slow convergence. Moreover, there is no proof of convergence, although it seems to perform well in practice. Some times it is possible that result may converge to local minimum and there is no way to reduce its possibility. Another problem is that of scaling, which may be handled using modular architectures and prior information about the problem.

ANN: Model Design & Training

Before applying ANN, the input data need to be standardized so as to fall in the range [0,1]. A typical hydrological variable, say river discharge (Q), which can vary between Qmin to some maximum value Q_{max} can be standardized by the following formula:

$$Q_s = \frac{Q - Q_{\min}}{Q_{\max} - Q_{\min}} \qquad (12)$$

where Q_s is the standardized discharge.

For a specific modeling problem, an ANN is designed in such a way to obtain a simple architecture which yields the desired performance. As there is no analytical solution to determine an optimal ANN architecture and therefore, a unique solution cannot be guaranteed. The numbers of input and output nodes are decided from the modeling problem. Further, the number of hidden layers and the number of nodes in each hidden layer are determined to produce most suitable ANN model architecture. Generally, a trial-and-error approach is used to find out the number of hidden layers and the number of nodes in each hidden layer. The number of nodes should be chosen carefully since the performance of a network critically depends on it. A network with too few nodes gives poor results, while it overfits the training data if too many nodes are present.

The primary goal of training is to minimize the error function by searching for a set of connection strengths and threshold values that cause the ANN to produce outputs that are equal or close to targets. There are different types of learning algorithms that are quite suitable for specific problems. The supervised training algorithm uses a large number of inputs and outputs patterns. The inputs are cause variables of a system and the outputs are the effect variables. This training procedure involves the iterative adjustment and optimization of connection weights and threshold values for each of nodes. In contrast, an unsupervised training algorithm uses only an input data set. The ANN adapts its connection weights to cluster input patterns into classes with similar properties. Supervised training is most common in water resources applications.

What Is Fuzzy Logic?

Fuzzy logic is a powerful problem-solving methodology with a myriad of applications in embedded control and information processing. Fuzzy provides a remarkably simple way to draw definite conclusions from vague, ambiguous or imprecise information. In a sense, fuzzy logic resembles human decision making with its ability to work from approximate data and find precise solutions.

Unlike classical logic which requires a deep understanding of a system, exact equations, and precise numeric values, Fuzzy logic incorporates an alternative way of thinking, which allows modeling complex systems using a higher level of abstraction originating from our knowledge and experience.

Fuzzy Logic allows expressing this knowledge with subjective concepts such as very hot, bright red, and a long time which are mapped into exact numeric ranges.

Fuzzy Logic has been gaining increasing acceptance during the past few years. There are over two thousand commercially available products using Fuzzy Logic, ranging from washing machines to high speed trains. Nearly every application can potentially realize some of the benefits of Fuzzy Logic, such as performance, simplicity, lower cost, and productivity.

Fuzzy Logic has been found to be very suitable for embedded control applications. Several manufacturers in the automotive industry are using fuzzy technology to improve quality and reduce development time. In aerospace, fuzzy enables very complex real time problems to be tackled using a simple approach. In consumer electronics, fuzzy improves time to market and helps reduce costs. In manufacturing, fuzzy is proven to be invaluable in increasing equipment efficiency and diagnosing malfunctions. Usefulness of fuzzy rule based modeling in hydrological modeling and forecasting is also demonstrated by various researchers.

Fuzzy Sets

A set *A* is a collection of objects belonging to a given universe *X*, where, for each possible object *x* from the universe *X*, it is decidable whether it belongs to the set *A* or not; if so, we say $x \in X$; if not, we say $x \notin X$. In ordinary (non fuzzy) set theory, elements either fully belong to a set or are fully excluded from it. The membership $\mu_{\square}(x)$ of a set *A* as a subset of the universe *X*, is defined as follows (for all $x \in X$):

$$\mu_A(x) = \begin{Bmatrix} 1, & \text{iff } x \in A \\ 0, & \text{iff } x \notin A \end{Bmatrix}$$

This means that an element is either a member of set $A(\mu_{\square}(x)=1)$ or not $(\mu_{\square}(x)=0)$. This strict classification is useful in the mathematics and other sciences. The idea behind fuzzy logic is to replace the set of truth values {0, 1} by the entire unit interval [0, 1]. A fuzzy set on a universe X is represented by a function which maps each element $x \in X$ to a degree of membership from the unit interval [0, 1]. These so-called membership functions are direct generalizations of characteristic functions. Figure 4 presents difference between boolean logic and fuzzy logic.

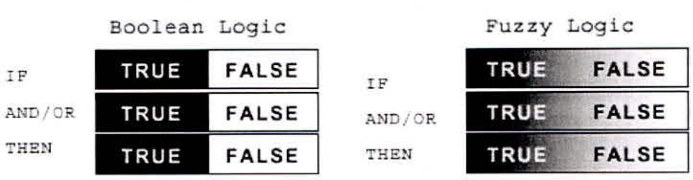

Figure 4: Boolean Logic Vs Fuzzy Logic

Membership Function Assignment and Rule Generation

First, partition the input and output space as small, medium, large etc. After partition, the next step is to assign a membership function. First the data points whose membership grades are among the highest are chosen. The mid-point of these data points is assigned grade of one, which is the index of membership function. Then a membership grade C $(0<C<1)$ is assigned.

The membership function is shown in the Figure 5, where c_{li} and b_{li} are the center and the half-width of the membership function respectively. And x is the average distance of the vertex to the left and the right edges. Thus, we have:

$$\frac{x}{b_{ij}} = \frac{1-C}{1} \Rightarrow b_{ij} = \frac{x}{1-C} \qquad \ldots(13)$$

C is a parameter to be assigned. This C is usually determined by experience, although some optimization techniques may be used. Typical values of CM vary from 0.5 to 0.8. After partitioning the input and output spaces and assigning the membership functions, the next step is to construct the rules.

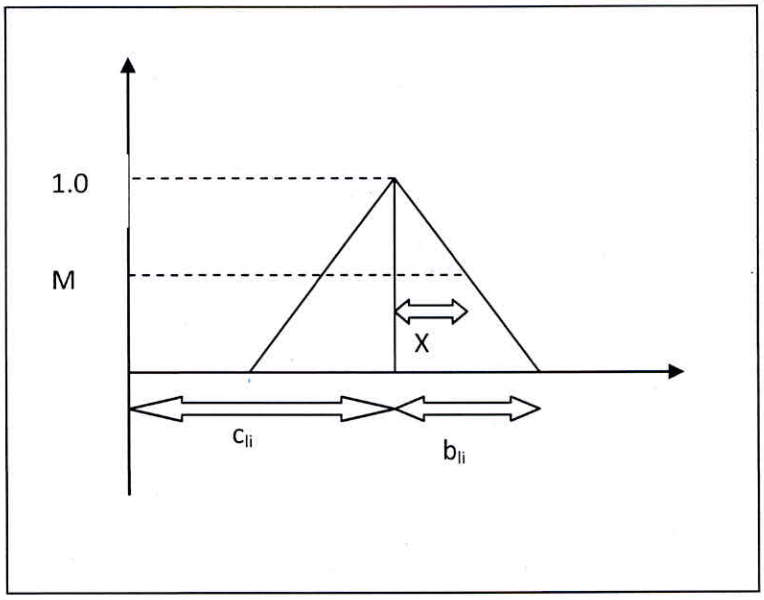

Figure 5: The Triangular Membership Function

Steps For Developing Fuzzy Logic Model

Step by step procedure for developing a fuzzy model is given below:

- Define the model objectives and criteria: What am I trying to model? What do I have to do to model the system? What kind of response do I need? What are the possible (probable) system failure modes?

- Determine the input and output relationships and choose a minimum number of variables for input to the Fuzzy Logic (FL) system.
- Using the rule-based structure of FL, break the modelling problem down into a series of IF X AND Y THEN Z rules that define the desired system output response for given system input conditions. The number and complexity of rules depends on the number of input parameters that are to be processed and the number of fuzzy variables associated with each parameter. If possible, use at least one variable and its time derivative. Although it is possible to use a single, instantaneous error parameter without knowing its rate of change, this cripples the system's ability to minimize overshoot for a step inputs.
- Create FL membership functions that define the meaning (values) of Input/Output terms used in the rules.
- Create the necessary pre- and post-processing FL
- Test the system, evaluate the results, tune the rules and membership functions, and retest until satisfactory results are obtained. Figure 6 presents steps involved for developing of fuzzy model.

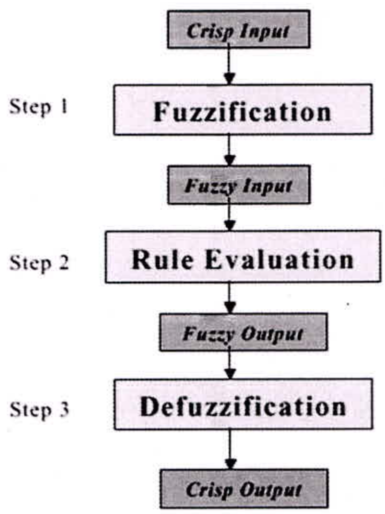

Figure 6: Steps for developing fuzzy model

Rule-Based Fuzzy Systems

In rule-based fuzzy systems, the relationships between variables are represented by means of fuzzy if–then rules in the following general form:

If antecedent proposition **then** consequent proposition.

Fuzzy propositions are statements like "x is big", where "big" is a *linguistic label*, defined by a fuzzy set on the universe of discourse of variable x. Linguistic labels are also referred to as fuzzy constants, fuzzy terms or fuzzy notions. Linguistic modifiers (hedges) can be used to modify the meaning of linguistic labels. For example, the linguistic modifier *very* can be used to change "x is big" to "x is *very* big".

The antecedent proposition is always a fuzzy proposition of the type "**x** is A" where **x** is a linguistic variable and A is a linguistic constant (term). On the basis of structure of the consequent proposition, different fuzzy rule based models are defined. In a *Linguistic fuzzy model* (Zadeh, 1973; Mamdani, 1977) both the antecedent and consequent are fuzzy propositions. *Singleton* fuzzy model is a special case where the consequents are singleton sets (real constants).

General Linguistic Fuzzy Model

The general Linguistic Fuzzy Model of a Multi-Input Single-Output system is interpreted by rules with multi-antecedent and single-consequent variables such as the following:

Rule l: IF I_1 is B_{l1} AND I_2 is B_{l2} AND I_r is B_{lr}

THEN O is D_l, $l = 1,2,..,n$ ...(14)

Where I_1 , I_2 ,..., I_r are input variables and O is the output, B_{ij} (i=1, ...,n, j=1,...,r) and D_i (i=1, ...,n) are fuzzy sets of the universes of discourse $X_1, X_2,.., X_r$, and Y of I_1 , I_2 ,..., I_r and O respectively. The above rule can be interpreted as a fuzzy implication relation:

$B_l = B_{l1} \text{ x } B_{l2} \text{ x.x } B_{lr} \rightarrow D_l$ in $(X = X_l \text{ x } X_2 \text{ x.x } X_r)\text{x}Y$:

$R_l(x,y) = T(B_l(x),D_l(y)), B_l(x)=T'(B_{l1}(x),B_{l2}(x),\ldots.B_{lr}(x))$ ...(15)

Where T and T' are the t-norm operators and may be different from each other. Let the fuzzy set A in the universe of discourse X be the input to the fuzzy system of (14). Then, each fuzzy IF-THEN rule determines a fuzzy set F_l in Y:

$F_l(y) = T(R_l(x,y),A(x))$ ...(16)

For a crisp input $x^* = (x_1^*, x_2^*,\ldots.x_r^*)$

$$A_i(x) = \begin{cases} 1, & \text{if } x_i = x_i^* \\ 0, & \text{if } x_i \neq x_i^* \end{cases} \quad \ldots(17)$$

Then

$F_l(y) = T(R_l(x,y),A(x))$

$= T(B_{il}(x),A(x),D_i(y))$ ...(18)

$= T(B_{il}(x^*),D_i(y))$

where $B_l(x)$ is called the Degree Of Firing (DOF) of rule l:

$B_l(x^*)=T'(B_{li}(x^*),B_{l2}(x_2^*),\ldots..B_{lr}(x_r^*))$ ...(19)

The output fuzzy set F of the fuzzy system is the t-conorm of the n fuzzy sets F_l $(l=1,2,\dots n)$:

$$F(y) = S[F_1(y), F_2(y),\dots F_n(y)] \qquad \dots(20)$$

Where, S denotes the t-conorm operator. To obtain a crisp value of the output, the commonly used Center of Area (COA) method, may be used.

$$y^* = \frac{\int_{y_0}^{y_l} yF(y)dy}{\int_{y_0}^{y_l} F(y)dy} \qquad \dots(21)$$

Where, the real interval $Y = [y_0, y_l]$ is the universe of discourse for the output.

The fuzzy system is usually not analytical, but analytical formulation is essential for the use of training algorithms like Back Propagation (BP) and Least Mean Squared (LMS). We, therefore, use the following simplified fuzzy inference system: First, T-norm and T-conorm operators are chosen to be the multiplication and addition operators, respectively. Then equation (20) becomes,

$$F(y) = \sum_{l=1}^{n} F_i(y) = \sum_{i=1}^{n} B_l(x^*)\cdot D_l(y) \qquad \dots(22)$$

Obviously, the summation brings the output fuzzy set $F(y)$ out of the unit interval. However, it does not have an effect on the defuzzified value. By substituting for $F(y)$ in (21) we get the COA defuzzified value:

$$y^* = \frac{\int_{y_0}^{y_1} y\sum_{l=1}^{n} B_l(x^*)D_l(y)dy}{\int_{y_0}^{y_1}\sum_{l=1}^{n} B_l(x^*)D_l(y)dy}$$

$$= \frac{\sum_{l=1}^{n} B_l(x^*)\left\{\frac{\int_{y_0}^{y_1} yD_l(y)dy}{\int_{y_0}^{y_1} D_l(y)dy}\right\}}{\sum_{l=1}^{n} B_l(x^*)}$$

$$= \frac{\sum_{l=1}^{n} B_l(x^*) y_l^*}{\sum_{l=1}^{n} B_l(x^*)} \qquad \ldots(23)$$

Where the y_l^* 's are the centroids of the fuzzy sets D_l.

The defuzzified value y^* is determined by the weighted average of the centroids of the individual consequent fuzzy sets. Using a symmetric triangular membership function, the fuzzy system becomes,

$$y^* = f(x) = \frac{\sum_{l=1}^{n} y_l^* (\prod_{l=}^{r} 1 - \frac{|x_l - c_{li}|}{b_{li}})}{\sum_{l=1}^{n} (\prod_{l=}^{r} 1 - \frac{|x_l - c_{li}|}{b_{li}})}, c_{li} - b_{li} \le x_i \le c_{li} + b_{li} \qquad \ldots(24)$$

Where, c_{li} and b_{li} are the center and the half-width of the triangular membership function

Ann & Fuzzy Logic in Groundwater Modelling and Management

The ability of an artificial neural network (ANN) to provide a data-driven approximation of the explicit relation between transmissivity and hydraulic head as described by the ground water flow equation is demonstrated by Garcia and Shigidi (2006). Nayak et al (2006) reported a research study that investigates the potential of artificial neural network technique in forecasting the groundwater level fluctuations in an unconfined coastal aquifer in India. The most appropriate set of input variables to the model are selected through a combination of domain knowledge and statistical analysis of the available data series. The results suggest that the model predictions are reasonably accurate as evaluated by various statistical indices. In general, the results suggest that the ANN models are able to forecast the water levels up to 4 months in advance reasonably well. Such forecasts may be useful in conjunctive use planning of groundwater and surface water in the coastal areas that help maintain the natural water table gradient to protect seawater intrusion or water logging condition.

Remarks

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture. Neural networks also contribute to hydrological modeling and forecasting. They are successfully used to model various hydrological processes. Even though neural networks have a huge potential one will only get the best of them when they are integrated with computing, AI, fuzzy logic and related subjects.

References

ASCE Task Committee on Application of Artificial. Neural Networks in Hydrology (2000a). Artificial Neural Networks in hydrology, I: Preliminary Concepts. Journal of Hydrologic Engineering, ASCE, 5(2), 115-123.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000b). Artificial Neurill Networks in hydrology, 11: Hydrological Applications. Journal of Hydrologic Engineering, ASCE, 5(2), 124-137.

Campolo, M., Soldati, A. & Andreussi, P. (2003) Artificial neural network approach to flood forecasting in the River Arno. Hydrol. Sci. J. 48(3), 381.398.

Chibanga, R., Berlamont, J. & Vandewalle, J. (2003) Modelling and forecasting of hydrological variables using artificial neural networks: the Kafue River sub-basin. *Hydrol. Sci. J.* 48(3), 363–379.

Cigizoglu, H. K. (2003) Estimation, forecasting and extrapolation of river flows by artificial neural networks. *Hydrol. Sci. J.* 48(3), 349–361.

Cigizoglu, H.K. & Kisi, O., (2005). Flow Prediction by Three Back Propagation Techniques Using k-fold Partitioning of Neural Network Training Data, *Nordic Hydrology*, 36(1), 49-64.

Clair, T.A. & Ehrman, J.M. (1998). Using neural networks to assess the influence of changing seasonal climates in modifying discharge, dissolved organic carbon, and nitrogen export in eastern Canadian rivers. *Water Resour. Res.* 34 (3), 447–455.

Coulibaly, P., Anctil, F. & Bobee, B. (1998). Real time neural network based forecasting system for hydropower reservoirs. In: Miresco, E.T. (Ed.), *Proceedings of the First International Conference on New Information Technologies for Decision Making in Civil Engineering*, October 10–13, University of Quebec, Montreal, Canada, pp. 1001–1011.

Dawson, C. W. & Wilby, R. L. (2001) Hydrological modeling using artificial neural networks. Progr. Phys. Geogr. 25(1), 80.108.

Garcia, L. A.and Shigidi, A. (2006) Using neural networks for parameter estimation in ground water. Journal of Hydrology, 318(1-4), 215-231.

Hu, T. S., Lam, K. C. & Ng, S. T. (2001) River flow time series prediction with range-dependent neural network. Hydrol. Sci. J. 46(5), 729.746.

James, W. (1890). *Psychology (Briefer Course)*. New York: Holt.

Jain, S.K., Das, D., and Srivastava, D.K. (1999). Application of ANN for reservoir inflow prediction and operation. *J. Water Resour. Planning Mgmt*, ASCE, 125(5), 263–271.

Jain, S. K. & Chalisgaonkar, D. (2000). "Setting up stage discharge relations using ANN." *J. Hydrologic Eng.*, ASCE, 5(4), 428–433.

Kisi, O., (2004a). River flow modeling using artificial neural networks, *J. of Hydrologic Engineering*, ASCE, Vol.9, No.1, 60-63.

Lin, G.F. & Chen, L.H. (2004). A non-linear rainfall-runoff model using radial basis function network, *J. of Hydrology*, 289, 1-8.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5*, 115 – 133.

Maier, H. R. & Dandy, G. C. (2000) Neural networks for the prediction and forecasting of water resources variables: a review of modeling issues and applications. Environ. Modelling & Software 15, 101.124.

Minns, A. W. & Hall, M. J. (1996) Artificial neural networks as rainfall runoff models. Hydrol. Sci. J. 4(3), 399.417.

Nayak, P.C., Satyaji Rao, Y. R. and Sudheer, K. P. (2006) Groundwater level forecasting in a shallow aquifer using artificial neural network approach. Water resources management, 20(1), 77-90.

Raman, H., and Chandramauli, V. (1996). Deriving a general operating policy for reservoir using Neural Network. Journal of Water Resources Planning and Management, 122(5), 342-347.

Raman, H. & Sunilkumar, N. (1995). Multivariate modelling of water resources time series using artificial neural networks", *Hydrol. Sci. J.* 40 (2), 145-163.

Rumelhart, D.E., McLelland, J.L. and the PDP Research Group, (1986). Parallel Distributed Processing, Explorations in the Micro Structure of Cognition, Vol. I: Foundations, The MIT Press, Cambridge.

Saad, M., Bigras, P., Turgeon, A. & Duquette, R. (1996). Fuzzy learning decomposition for the scheduling of hydroelectric power systems. *Water Resour. Res.*, 32 (1), 179–186.

Shamseldin, A. Y., O.Connor, K. M. & Liang, G. C. (1997) Methods for combining the outputs of different rainfall runoff models. J. Hydrol. 197, 203.229.

Smith, J. & Eli, R.N. (1995). Neural network models of rainfall–runoff process. *J. Wat. Resour. Plan. Mgmt*, 499–508.

Solomatine, D. P. & Dulal, K. N. (2003) Model trees as an alternative to neural networks in rainfall–runoff modelling. *Hydrol. Sci.* J. 48(3), 399–411.

Sudheer, K.P. & Jain, S.K. (2003). Radial Basis Function Neural Networks for Modeling Rating Curves, *J. Hydrologic Eng.*, ASCE, 8(3), 161–164.

Sugawara, M. (1979). Automatic calibration of tank model. Hydrological Science Bulletin, 24(3).

Supharatid, S. (2003). Application of a neural network model in establishing a stage-discharge relationship for a tidal river, *Hydrological Processes*, 17, 3085-3099.

Tayfur, G. (2002) Artificial neural networks for sheet sediment transport. *Hydrol. Sci. J.* 47(6), 879–892.

Tokar, A.S. & Johnson, P.A. (1999). Rainfall–runoff modeling using artificial neural networks. *J. of Hydrol. Eng,* ASCE, 4(3), 232–239, 1999.

Wilby, R. L., Abrahart, R. J. & Dawson, C. W. (2003) Detection of conceptual model rainfall runoff processes inside an artificial neural network. Hydrol. Sci. J. 48(2), 163.181.

Zhang, B. & Govindaraju, R. S. (2000) Prediction of watershed runoff using Bayesian concepts and modular neural networks. Water Resour. Res. 36(3), 753.763.