A hybrid linear-neural model for river flow forecasting

M. Chetan[1] and K. P. Sudheer[1]

Received 28 February 2005; revised 21 September 2005; accepted 23 January 2006; published 1 April 2006.

[1] This paper presents a novel hybrid linear-neural (LN) model formulation to effectively model rainfall-runoff processes. The central idea of the proposed model framework is that the hidden layer of an artificial neural network (ANN) model be designed with a combination of linear and nonlinear neurons. A training algorithm for the proposed model is designed based on minimum description length criteria. The advantage of the algorithm is that the final architecture of the LN model is arrived at during the training process, thus avoiding selection from a class of models. The proposed model has been developed and evaluated for its performance for forecasting the river flow of Kolar basin, in India. The values of three performance evaluation criteria, namely, the coefficient of efficiency, the root-mean-square error, and the coefficient of correlation, were found to be very good and consistent for flows forecasted 1 hour in advance by the LN model. The value of the relative error in peak flow prediction was within reasonable limits for the model. The forecasts by the LN model at higher lead times (up to 6 hours) are found to be good. A relative evaluation of LN model performance with that of an ANN model and of a multiple linear regression model indicates that the LN model effectively combines the strength of the other two, implying that the LN model seems to be well suited to exploit the information to model the nonlinear dynamics of the rainfall-runoff process.

Citation: Chetan, M., and K. P. Sudheer (2006), A hybrid linear-neural model for river flow forecasting, *Water Resour. Res.*, *42*, W04402, doi:10.1029/2005WR004072.

1. Introduction

[2] River flow forecasting has always been one of the most important issues in hydrology. Forecasting a river flow provides a warning of impending stages during floods and assists in regulating reservoir outflows during low river flows for water resources management. To date, a wide variety of rainfall-runoff models have been developed and applied for flood forecasting. Most of these models have been developed based on either physical considerations or a system theoretic approach. In the physical approach, the primary motivation is the study of physical phenomena and their understanding, while in the system theoretic approach the concern is with the system operation, not the nature of the system by itself or the physical laws governing its operation. A limitation of the physical approach is that even though the complexity of the process can be perceived easily, its representation in terms of mathematical description suitable for quantitative prediction is much more difficult. It will always involve important simplification and approximations. As a result, there exists some skepticism in the use of physically based models for predicting watershed runoff [*Grayson et al.*, 1992]. On the other hand, the system theoretic approach considers the catchment as a "black box," without any reference to the internal processes that control the transformation of rainfall to runoff; they are empirical in nature. The need for a system theoretic approach arises in many instances, primarily from the complexities inherent in the physical approach and a belief that the system behavior may be approximated by major physical factors, thus permitting the neglect of minor features and exact spatial features of the physical parameters [*Sudheer*, 2000]. While such models do not provide any knowledge about the physics of the hydrologic processes, they are, in particular, very useful for river flow forecasting where the main concern is with making accurate predictions of flow at specific watershed locations [*Hsu et al.*, 1995].

[3] In the black box modeling approach, empirical relationships between the input (rainfall and/or any exogenous variables) and the output (runoff) have been widely used in hydrologic prediction. However, most of them assumed a linear relationship between the variables presumably because of the simple procedures for parameter estimation [*Sefton and Howarth*, 1998], while it is clearly understood that the modeled relationship is highly nonlinear. A similar argument holds well with the autoregressive moving average model with exogenous inputs (ARMAX) of *Box and Jenkins* [1976]. The major concern with nonlinear black box models is structure identification and parameter estimation. As a consequence, there are very few truly nonlinear system theoretic models [e.g., *Jacoby*, 1966; *Amorocho and Brandstetter*, 1971; *Ikeda et al.*, 1976]. In most of them, linearity or piecewise linearity has been assumed [*Natale and Todini*, 1976; *Hsu et. al.*, 1995]. Furthermore, most of

[1]Department of Civil Engineering, Indian Institute of Technology Madras, Chennai, India.

the nonlinear models used a predefined structure with unknown parameters, which is likely to misrepresent the process unless carefully designed. In this context, data-driven models, which can discover relationships from input-output data without having the complete physical understanding of the system, may be preferable. The data-driven models aim primarily at establishing functional relationships between input and output variables without defining a functional relationship a priori. For data-driven modeling, the most widely employed tool in the recent past is the artificial neural network (ANN) technique. This technique has been rigorously applied to rainfall-runoff modeling [*Hsu et al.*, 1995; *Sajikumar and Thandaveswara*, 1999; *Sudheer et al.*, 2002, 2003], and the results have been highly promising. The reason for such an increasing interest resides in their intrinsic generality, flexibility, and global performance in most applications where other models either tend to fail or become cumbersome [*Shamseldin et al.*, 2002].

[4] Besides the fundamental question of choosing the right set of input variables, defining an adequate neural topology for approximating a function by a neural network still remains an unsatisfactorily solved question. The search for a satisfactory compromise between good data fitting on one side and good generalization properties on the other side has oriented the design of several online and off-line model building procedures. Among online techniques, constructive algorithms, as, for example, cascade correlation [*Kwok and Yeung*, 1997; *Karunanithi et al.*, 1994], follow an incremental approach by starting with a small network and trying to increase it step by step. On the other hand, pruning algorithms, as optimal brain damage [*Le Cun et al.*, 1990] to optimal brain surgeon [*Hassibi et al.*, 1994], follow a decremental approach by starting with a large network and trying to eliminate unnecessary connections [*Reed*, 1993]. Both types of step-by-step evolution, however, do not ensure the reaching of the best topology [*Sudheer*, 2000]. In hydrological applications, the number of hidden neurons, which is responsible for capturing the dynamic and complex relationship between various input and output variable, is often arrived at after a long trial-and-error procedure.

[5] It is a common belief that the ANN models of the rainfall-runoff process (or any other process) are purely black box models as they do not explain the process being modeled, but for a few recent studies [*Wilby et al.*, 2003; *Jain et al.*, 2004; *Sudheer*, 2005]. However, it must be realized that the hydrometeorological data that are employed in developing rainfall-runoff models (ANN or physics based) contain important information about the physical process being modeled, and this information gets embedded or captured inside the model. *Jain et al.* [2004] illustrate that a trained ANN captures different components of the physical process being modeled through its massively parallel and distributed architecture, and they demonstrate that the hidden neurons in the ANN rainfall-runoff model approximate various components of the hydrologic system.

[6] It appears that constructing models from time series with nontrivial dynamics involves the problem of how to choose the best model from within a class of model, or to choose between competing classes. In ANN modeling, it is assumed that there is a random process generating $(\mathbf{Y_t}, \mathbf{X_t}) \in \mathrm{RXR}^d$, $t \in Z$, for which there is a relationship

$$\mathrm{Y_t} = F(\mathbf{X_t}) + \varepsilon_t \tag{1}$$

where the random variables ε_t are independent of $\mathbf{Y_r}$ and $\mathbf{X_s}$ for $r \geq t$ and $s > t$. Building a model means, given some realizations of this process $(\mathrm{y_t}, \mathrm{x_t})$, $t = 1,\ldots, n$, find an approximation $\hat{F}$ of F. Typically, when modeling a time series, one restricts attention to a subclass of models $\hat{F}(\mathbf{X}) = \mathrm{G}(\mathbf{X}, \lambda)$, parameterized by $\lambda \in R^m$ for some m. The potential number of m parameters may be infinite for a class of models. For linear models, for example, autoregressive (AR) models, the parameters are potentially infinite in number, but the selection problem is relatively easy. One fits models AR(k) of increasing order, $k = 0, 1, 2, 3, \ldots$, until it is found that increasing the order of the model makes no significant improvement. In the case of nonlinear models, however, there are unaccountably many possible subclasses to choose from. Unless one has some prior knowledge about the time series, derived from the physics of the phenomenon being studied, he may find it difficult to select a subclass to work with.

[7] The ANN modeling approach is based on the concept of reconstruction of a single-variable series in a multidimensional phase space to represent the underlying dynamics. An ANN uses local approximation through ridge functions while modeling the function being studied [*Sudheer et al.*, 2003]. Hence it appears that the number of clearly distinct clusters present in the multidimensional phase space can be considered as a guideline to approximate the number of hidden neurons in the architecture. Nonetheless, it needs to be verified through a model selection procedure. Also, following *Jain et al.* [2004], who have illustrated that the hidden neurons in an ANN model approximate various components of the rainfall-runoff process, it seems that all the hidden neurons in the final ANN architecture need not be always nonlinear (i.e., the transfer function of some of the hidden neurons need not be a nonlinear function) because some of the subprocesses in a basin may show a linear variation along the time (for example, the interflow portion on the recession limb of a hydrograph). In view these heuristics, a research study has been formulated to develop and test a hybrid linear-neural (LN) model that consists of linear and nonlinear hidden neurons in the hidden layer of an ANN for rainfall-runoff modeling, which can be used for river flow forecasting. The central idea is to consider a new formulation that combines the idea from multiple linear regression models and from artificial neural networks.

[8] The major objective of this paper is to illustrate the development of a hybrid LN model for river flow forecasting. The paper also focuses on the issue of arriving at an appropriate network architecture to model the process. More specifically, the paper discusses the concepts behind the proposed hybrid model and the algorithm that is developed for training the hybrid model. The proposed modeling approach is evaluated on a real-world case study of an Indian River basin, and the results are discussed in detail. The paper is organized as follows: Following the introduction, the background, theoretical considerations, and the algorithm for training of the proposed LN model are

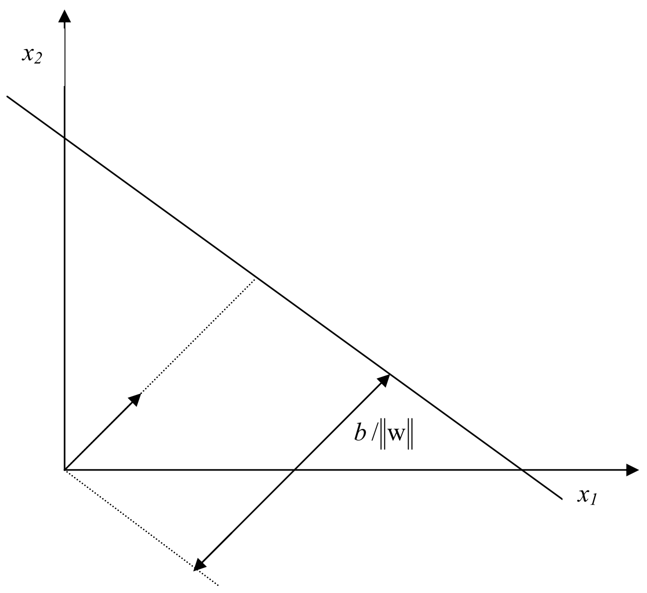


Figure 1. Hyperplane defined by $\mathbf{w}'\mathbf{x} = b$ in two-dimensional space.

discussed. In the succeeding sections, the case study and performance of the LN model on the selected case are presented and discussed in detail. The article ends with the conclusions drawn from this study.

2. Hybrid Linear-Neural Model

2.1. Background

[9] A flow hydrograph, which is normally used as the output variable in an ANN rainfall runoff model, consists of various components that result from different physical processes in a watershed. For example, the rising limb of a runoff hydrograph is the result of the gradual release of water from various storage elements of a watershed due to gradual repletion of the storage due to the rainfall input. The rising limb of the hydrograph is influenced by varying infiltration capacities, watershed storage characteristics, and the nature of the input, i.e., intensity and duration of the rainfall, and not so much by the climatic factors such as temperature and evapotranspiration, etc. [*Zhang and Govindaraju*, 2000]. On the other hand, the falling limb of a hydrograph is the result of the gradual release of water from various storages of the watershed after the rainfall input has stopped and is influenced more by the storage characteristics of the watershed and climatic characteristics to some extent. Further, the falling limb of a flow hydrograph can be divided into three parts: initial portion just after the peak, middle portion, and the final portion. The initial portion of the falling limb of a flow hydrograph is influenced more by the quick flow (or interflow), the middle portion of the falling limb is more dominated by the delayed surface flow, and the final portion of the falling limb (of smaller magnitudes) is dominated by the base flow.

[10] While explaining the internal behavior of an ANN rainfall runoff model, *Sudheer and Jain* [2004] indicate that individual hidden neurons represent the dynamics of different portions of the hydrograph. Complementing to this, *Sudheer* [2005], based on an attempt to extract knowledge from trained ANN models, reports that each variable in the input vector influences the shape of the hydrograph in different ways. This varied influence is accomplished through the dynamics of hidden neurons in the ANN. It follows that if there is a region of the function being mapped (hydrograph in the current study), where the function behaves linearly in the multidimensional phase space, that region can be mapped by a linear neuron effectively. It is well understood that the portion of the hydrograph which represent the interflow varies almost linearly with time. Consequently, it can be concluded that some of the hidden neurons in the ANN rainfall runoff model can be linear neurons. Hence an ANN model that contains linear and nonlinear hidden neurons will be able to effectively represent rainfall-runoff processes. However, a decision on how many of such linear (and/or nonlinear) neurons are required and which are their corresponding input neurons is not trivial. This decision has to be established though a model selection procedure.

2.2. Theoretical Considerations

[11] Consider the output $f_{(\mathrm{w},b)}$ of one neuron of the hidden layer of a neural network with logistic activation function (which is the commonly employed transfer function) expressed as

$$f_{(\mathrm{w},b)}(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}'\mathbf{x} + b)}, \tag{2}$$

where $\mathbf{x}$ is an n-dimensional input vector, $\mathbf{w} = [w_1, \ldots, w_n]$ is the vector of weights of the incoming signals at the considered neuron, and b is the bias parameter of the same neuron. When $\mathbf{w}'\mathbf{x} = b$, the parameters $\mathbf{w}$ and b define a hyperplane in n-dimensional Euclidian space,

$$\mathrm{H} = \{\mathbf{x} \in \mathrm{R}^{\mathrm{n}} | \mathbf{w}'\mathbf{x} = b\}. \tag{3}$$

Figure 1 shows an example of hyperplane in two-dimensional space. The direction of $\mathbf{w}$ determines the orientation of the hyperplane and the scalar term $b/\|\mathbf{w}\|$ determines the position of the hyperplane in terms of its distance from the origin. A hyperplane induces a partition of the space into two regions defined by the half spaces

$$\mathrm{H}^{+} = \{\mathbf{x} \in \mathrm{R}^{\mathrm{n}} | \mathbf{w}'\mathbf{x} \geq b\} \tag{4}$$

$$\mathrm{H}^{-} = \{\mathbf{x} \in \mathrm{R}^{\mathrm{n}} | \mathbf{w}'\mathbf{x} < b\} \tag{5}$$

associated to the states, activated or not, of the neuron. With h hyperplanes, an n-dimensional space will be split into several polyhedral regions. Each region is defined by the nonempty intersections of the half spaces (4) and (5) of the hyperplane. When a linear function is used at the hidden neuron, the neuron is always at the activated state (if the neuron is without a bias term), and the activation will be equal to $\mathbf{w}'\mathbf{x}$. The linear neuron does not constraint splitting of the n-dimensional space into polyhedral regions, but the regions may become narrowed (since there is no transformation applied to the input signal), which in turn may help a better local approximation.

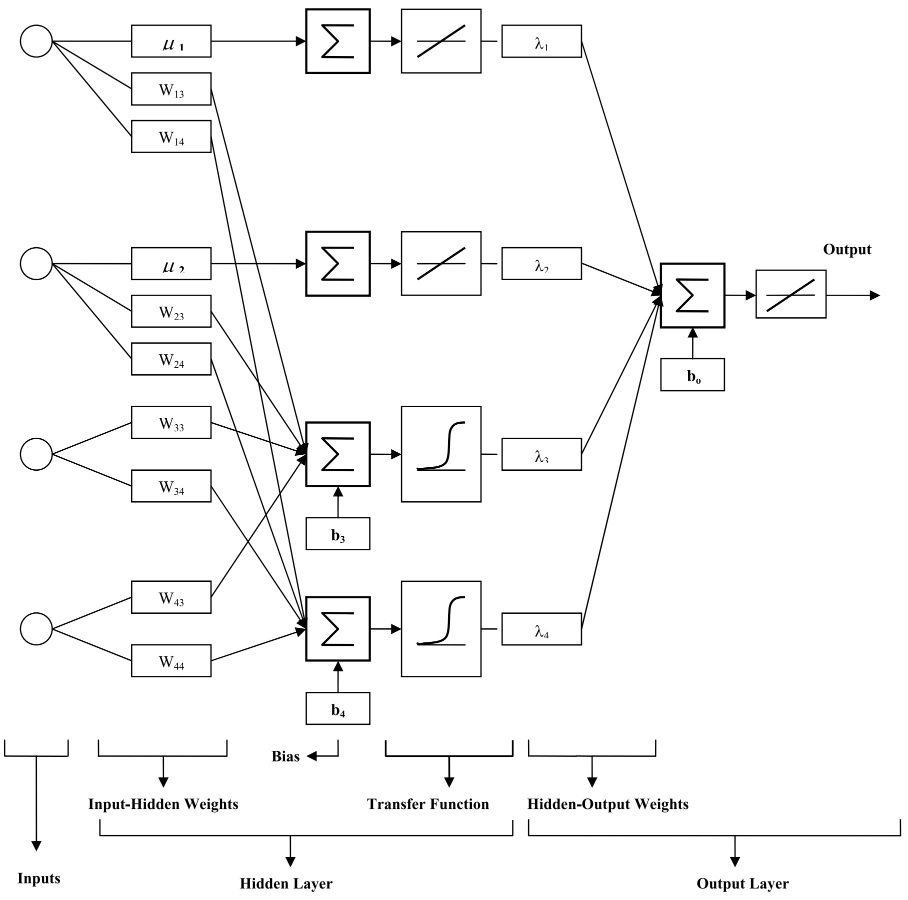


Figure 2. General architecture of the LN model.

[12] As stated earlier, the main idea of the proposed LN model is to use ANN to create a smooth multidimensional threshold structure with a combination of linear and nonlinear hidden neurons. Suppose that an n-dimensional space is spanned by a vector **x** formed by a lagged observations of a time series y_t (river flow) and/or any other exogenous variables (rainfall, evaporation, etc.), and suppose there are h_1 nonlinear hidden neurons whose output is defined by $f_{(\mathrm{w}_i, b_i)}$ (x), i = 1,..., h_1, and there are h_2 linear hidden neurons whose output is defined by $g_{(\mu_j)}(x_j) = \mu_j . x_j$ $j = 1, \ldots, h_2 | h_2 < n$, each of which defines a smooth multivariate threshold. Mathematically, the model can be expressed as

$$y_t = \sum_{i=1}^{h_1} \lambda_i f_{\left(\mathrm{w}_i' \mathrm{x}, \mathrm{b}\right)} + \sum_{j=1}^{h_2} \lambda_j g_{\left(\mu_j\right)}\left(x_j\right) \tag{6}$$

in which λ is the hidden-output weights and all the other variables are as defined above. The architecture of the proposed linear-neural model is presented in Figure 2, in which two linear and two nonlinear hidden neurons are considered in the structure. Note that the linear neurons do not get signal from all the input variables, and the input node from where it get the signal is determined during the training, which is explained in the next section. It may also be noted that the transfer function at the output layer of the LN model is considered to be linear, which has two advantages: (1) It avoids the problem of saturation at the output neuron and (2) the weights of the hidden-output neurons can be obtained using the linear least squares algorithm.

2.3. Training Linear-Neural Model

[13] The architecture of the linear-neural (LN) model as visualized in Figure 2 is quite complex in terms of parameter optimization. The parameters that need to be optimized in the LN model are the weight and bias parameters of the nonlinear neurons, weight parameters of the linear neurons, and the output weight parameters. Also, the number of linear nodes and their counterpart in the input vector, and the number of nonlinear neurons is not known a priori. If one wishes to use a back propagation algorithm (or any other training algorithm), one needs to fix the architecture first and has to arrive at the best architecture through trial and error. Instead, one can utilize a model selection procedure which will select the appropriate number of linear and nonlinear nodes during training. In the current study, we employed the minimum description length based model selection procedure [*Small and Tse*, 2002] to train the LN model.

2.4. Minimum Description Length Criterion

[14] The minimum description length (MDL) principle [*Rissanen*, 1978] is a method for inductive inference that provides a generic solution to the model selection problem. The MDL is a trade-off between the model complexity and the "goodness of fit" by avoiding too simple models and overly complex models. MDL procedures automatically and

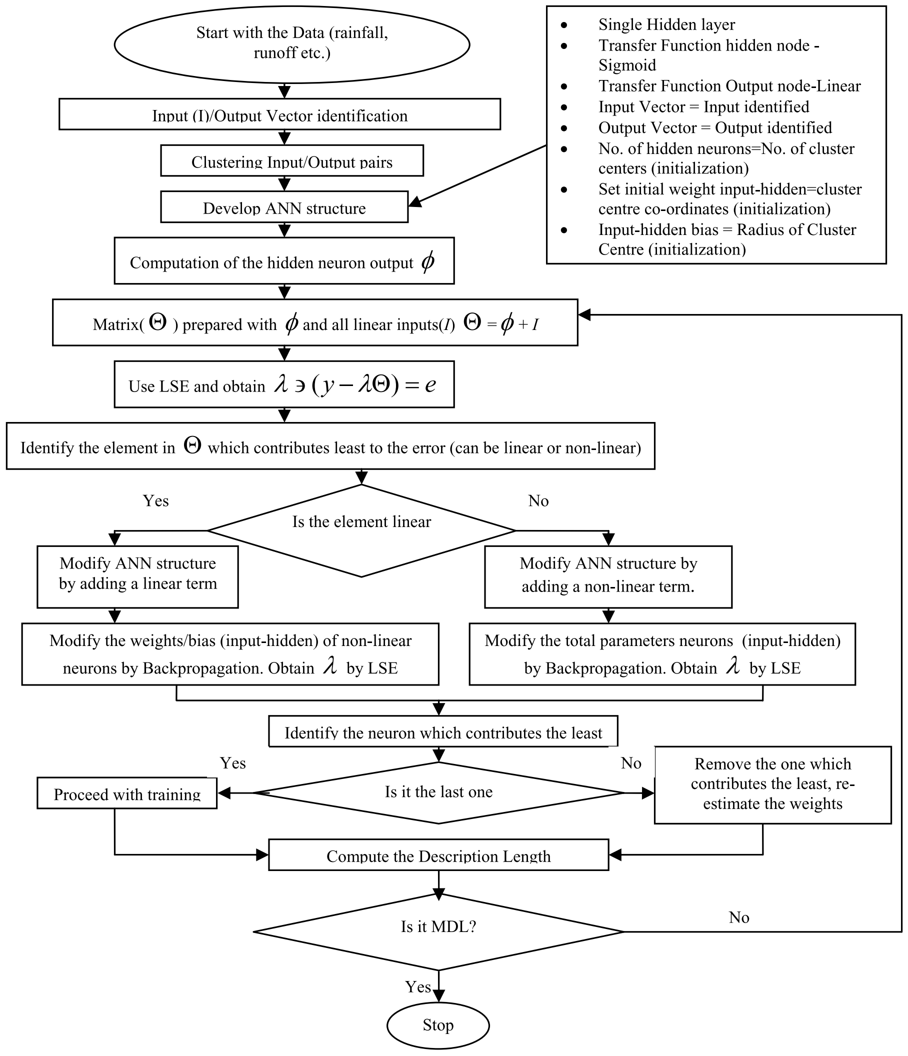

Start with the Data (rainfall, runoff etc.)
Input (I)/Output Vector identification
Clustering Input/Output pairs
Develop ANN structure
Single Hidden layer
Transfer Function hidden node - Sigmoid
Transfer Function Output node-Linear
Input Vector = Input identified
Output Vector = Output identified
No. of hidden neurons=No. of cluster centers (initialization)
Set initial weight input-hidden=cluster centre co-ordinates (initialization)
Input-hidden bias = Radius of Cluster Centre (initialization)
Computation of the hidden neuron output ϕ
Matrix(Θ) prepared with ϕ and all linear inputs(I) $\Theta = \phi + I$
Use LSE and obtain $\lambda \ni (y - \lambda\Theta) = e$
Identify the element in Θ which contributes least to the error (can be linear or non-linear)
Is the element linear
Yes
No
Modify ANN structure by adding a linear term
Modify ANN structure by adding a non-linear term.
Modify the weights/bias (input-hidden) of non-linear neurons by Backpropagation. Obtain λ by LSE
Modify the total parameters neurons (input-hidden) by Backpropagation. Obtain λ by LSE
Identify the neuron which contributes the least
Is it the last one
Yes
No
Proceed with training
Remove the one which contributes the least, re-estimate the weights
Compute the Description Length
Is it MDL?
No
Yes
Stop

Figure 3. Flowchart for the algorithm.

inherently protect against over-fitting and can be used to estimate both the parameters and the structure (e.g., number of parameters) of a model. The MDL principle considers that a model that provides the most compact description of a time series is the best.

[15] Consider a model $\hat{F}$ (**X**) = ***G***(**X**, λ), parameterized by $\lambda \in R^m$ for some m that is proposed to fit the data. The model can be described by describing its coefficients in a certain precision (number of bits per parameter). Thus the higher the degree of the model or the precision of the parameters, the more bits we need to describe it and the more complex it becomes. It follows that the better the model fits the data, the shorter the description of the model will be. A typically very complex model (large number of parameters and higher order) fits the data very well, while one can find a simple model also that fits the data badly. Hence an appropriate model, which has good generalization properties, has to be selected by a trade-off between the model complexity (in terms of the parameters) and the goodness of fit. Hence the sum of the description lengths of model (in terms of its parameter) and the model error when minimized would yield a good model that fits the data well.

[16] Hence description length of a time series $D(k)$ is the sum of the model description length $M(k)$, of the model of size k (number of parameters), and the description length of the model prediction errors $E(k)$. As the model size k increases, $E(k)$ decreases while $M(k)$ increases. According to the MDL principle, the optimal model size is that which minimizes the sum $D(k) = M(k) + E(k)$. Hence it appears that if the computation of the description length of the model is incorporated in the training process, the optimal model that represents the time series in question can be decided effectively. The procedure for computing the description length of a model is described in Appendix A.

2.5. Training Algorithm

[17] The major task in training the LN model is identifying the number of linear and nonlinear neurons and determining the values of model parameters. An initial guess for the number of nonlinear neurons can be made by a clustering algorithm that provides clearly distinct clusters in the multidimensional input space. This initial guess is valid since the ANN technique is based on local approximation [*Sudheer*, 2005]. In the current study, the subtractive clustering algorithm [*Chiu*, 1994] has been employed for clustering the input space. The training algorithm starts with only h_1 nonlinear neurons (set initially equal to the number of clusters) in the model. The initialization of parameters of

these nonlinear neurons is done by setting them equal to the coordinates of the cluster centers. The bias for these neurons is taken initially equal to the effective cluster radius for each cluster. The hidden-output weight vector (λ) can be estimated using linear least squares error algorithm (LSE), and the error (e) for the current model can be computed. Now a combination of all possible linear variables and the current hidden neuron output ($\Theta^{(T)}$) is formulated as a temporary vector (we call this vector a basis vector), from which the basis variable that fits the current error the best is identified. The identified basis variable can be a linear or a nonlinear variable, based on which the number of linear or nonlinear neurons in the hidden layer is changed, and a modified architecture for the LN model is framed. Subsequently, modify the parameters of the nonlinear neurons using back propagation algorithm, and then estimate the hidden-output weights using LSE. At this stage, identify the node in the current hidden neurons (linear + nonlinear) that contributes the least, and if this is the last neuron added to the model, then enlarge the model; otherwise remove the neuron that does the least. The description length of the model is computed after reestimating the model parameters. The process is repeated until the minimum description length is reached. A flowchart of the training algorithm is presented in Figure 3.

[18] The training algorithm can be described mathematically, as follows:

[19] 1. Let $\Theta^{(n)} = \mathbf{x}$ be the set of all possible linear terms (n-dimensional input vector) and let there be N patterns available for training. (Note that $\Theta^{(n)}$ is an "n X N" matrix, considering all N patterns.) The vector $\mathbf{x}$ can be determined using any procedure that selects the best possible combination of influencing variables [e.g., *Sudheer et al.*, 2003].

[20] 2. Generate a set of candidate p hidden nonlinear neurons $\Theta^{(p)}$ such that $\Theta^{(p)} \subseteq \{(\mathbf{x}.\mathbf{w}_{\mathrm{p}} - b_p) | \mathbf{w} \subseteq \mathrm{R}^{\mathrm{n}}, b \subseteq \mathrm{R}$ (i.e., choose a set of candidate weight vector (**w**) and bias (b) for each neuron. Initially, set p equal to the number of clusters derived from any clustering algorithm, and set **w** equal to the coordinates of the cluster center and b equal to the effective radius of respective cluster center. Note that weights (**w**) and bias (b) can be initialized randomly also).

[21] 3. Evaluate the transfer function on the nonlinear neurons to derive $\Phi^{(p)}$. Assuming sigmoid function, $\Phi^{(p)} = 1/(1 + \exp(-\Theta^{(p)}))$.

[22] 4. Compute the parameters $\Delta_p = [\lambda_1, \lambda_2, \ldots, \lambda_p]$ such that the error $\mathbf{e} = \mathbf{y} - \Delta_p \Phi^{(p)}$ is minimal (using least squares error (LSE) algorithm).

[23] 5. Select the row $\theta \subseteq \Theta^{(T)} = \Theta^{(n)} \cup \Theta^{(p)}$, where $\Theta^{(T)}$ is a "$(p + n)$ X N" matrix for total patterns such that $|\sum_{j=1}^{N} \theta_i (\boldsymbol{y}_j).\boldsymbol{e}_j|$ is maximal, where $1 \leq i \leq (p + n)$ (i.e., choose the basis variable/function that fits the current error best). Note that here $\theta_i(\boldsymbol{y}_j)$ implies the jth element in ith row of $\Theta^{(T)}$.

[24] 6. Let

$$\Phi^{(p+1)} = \begin{bmatrix} \Phi^{(p)} \\ \phi_{(p+1)} \end{bmatrix},$$

where $\phi_{(p+1)}$, is the evaluation of the row vector θ, which is identified at step 5, and $\Phi^{(p)}$ is p X N matrix, making $\Phi^{(p+1)}$ a "$p + 1$ X N" matrix. Here, if the vector θ corresponds to a nonlinear neuron (implying that the original ANN with p nonlinear hidden neurons gets modified to an ANN having

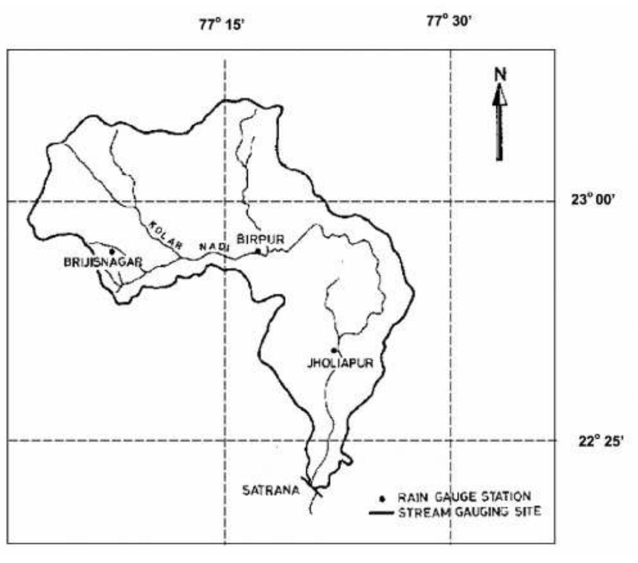


Figure 4. The basin map of Kolar River, India (up to Satrana gauging site).

(p + 1) nonlinear hidden neurons), perform the error back propagation algorithm to estimate the weights and bias for the hidden-input connections of the modified ANN structure. Then evaluate the transfer function over "p + 1" hidden neurons and set $\Phi^{(p+1)}$ equal to the evaluated values. On the other hand, if the vector θ corresponds to a linear neuron, set the last row in $\Phi^{(p+1)}$ equal to the value of the identified vector θ.

[25] 7. Compute the parameters Δ_{p+1} such that $\boldsymbol{e} = \mathbf{y} - \Delta_{p+1}\Phi^{(\mathbf{p+1})}$ is minimal.

[26] 8. Given

$$\Phi^{(p+1)} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ . \\ . \\ \Phi_{p+1} \end{bmatrix},$$

find i $(1 \leq i \leq p + 1)$ such that $|\sum_{l=1}^{N} \phi_i (y_l).e_l| < |\sum_{l=1}^{N} \phi_j (y_l).e_l|$ for all l $(1 \leq l \leq N)$ (i.e., find the term in the current model that contributes the least). Note that here ϕ_i $(\mathbf{y}_l)$ implies the lth element in ith row of $\Phi^{(p+1)}$.

[27] 9. If $i = p + 1$, then keep the p + 1th neuron; otherwise set

$$\Phi_p = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ . \\ . \\ \Phi_{i-1} \\ \Phi_{i+1} \\ . \\ . \\ \Phi_{p+1} \end{bmatrix},$$

where ϕ_i is the ith row of the $(p + 1)$ X N matrix Φ_{p+1} (i.e., if the last neuron added now contributes the least, then enlarge

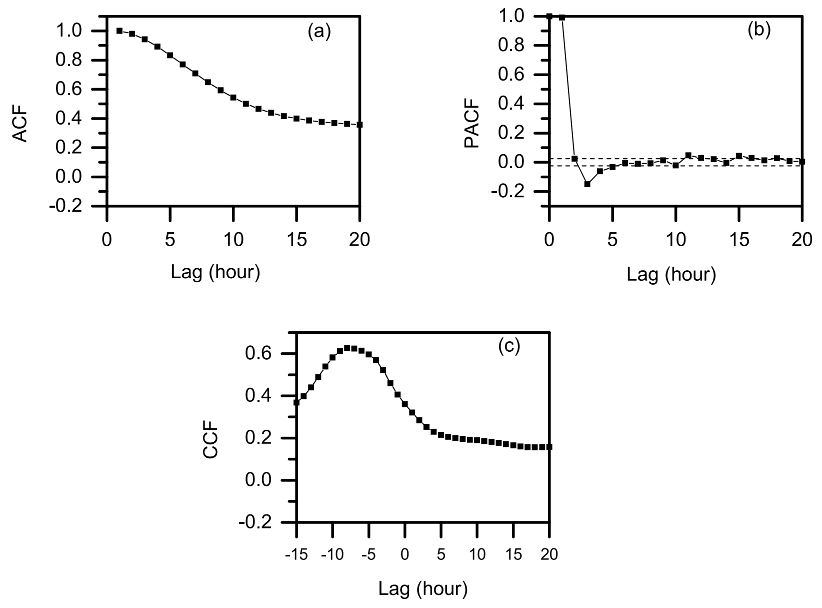


Figure 5. Correlogram plots of the data series: (a) autocorrelation function (ACF); (b) partial autocorrelation function (PACF); (c) cross-correlation function (CCF) between runoff at different lags of rainfall.

the model; otherwise remove the hidden neuron that does the least).

[28] 10. Reestimate the weights and bias of the nonlinear neurons (**w** and b) (of the modified architecture of ANN at step 9) using the back propagation algorithm, and then recompute the parameters Δ_p and the model prediction errors $\boldsymbol{e}$.

[29] 11. Compute the description length of the model (Appendix A). Repeat steps 5–10 till a minimum description length is reached.

[30] The major advantage of this algorithm is that the parameters of the candidate neurons are recomputed for each model expansion. The least squares error estimates of the output layer weights are computed in three different places in this algorithm (steps 4, 7, and 10) and for each value of p. Step 5 selects from the current host of candidates, the best fit to the current error, and step 9 rejects the current worst neuron in the model. Only when the neuron selected in steps 5 and 9 differ does the model expand.

3. Application

3.1. Study Area and Data Sets

[31] The proposed LN model has been applied to river flow forecasting of the Kolar River, in India, on an hourly time step. The Kolar River is a tributary of the river Narmada that drains an area of about 1350 km^2 before its confluence with Narmada near Neelkanth. In the present study the catchment area up to the Satrana gauging site is considered (Figure 4), which constitutes an area of 903.87 km^2. The 75.3-km-long river course lies between north latitude 22°40′ to 23°08′ and east longitude 77°01′ to 77°29′. Topographically, the Kolar subbasin can be divided into two zones. The upper four fifths, having elevations

Table 1. Performance Evaluation Criteria[a]

Evaluation Criteria	Definition
Coefficient of correlation (R)	$R = \left[\left(\sum_{i=1}^{n}\left(y_i^o - \overline{y^o}\right)\left(y_i^c - \overline{y^c}\right)\right) \Big/ \left(\sqrt{\sum_{i=1}^{n}\left(y_i^o - \overline{y^o}\right)^2}\sqrt{\sum_{i=1}^{n}\left(y_i^c - \overline{y^c}\right)^2}\right)\right]$
Coefficient of efficiency (E)	$E = 1 - \sum_{i=1}^{n}\left(y_i^o - y_i^c\right)^2 \Big/ \sum_{i=1}^{n}\left(y_i^o - \overline{y^o}\right)^2$
Root-mean-square error (RMSE)	$RMSE = \sqrt{\sum_{i=1}^{n}\left(y_i^o - y_i^c\right)^2 \Big/ n}$
Standard error of estimate (SEE)	$SEE = \sqrt{\left(\sum_{i=1}^{n}\left(y_i^o - y_i^c\right)^2\right) \Big/ \nu}$
Noise-to-signal ratio	NS = SEE/σ_y

[a]Here y_i^o and y_i^c are the observed and computed flow values, respectively, at time t; $\overline{y^o}$ and $\overline{y^c}$ are the mean of the observed and computed flow values corresponding to n patterns; ν is the number of degrees of freedom; and σ_y is the standard deviation of the observed flow. Normalized values of these statistics are obtained by dividing the value by the observed mean.

Table 2. Coordinates of the Cluster Centers of the Input Space and the Corresponding Output (From Subtractive Clustering)

Input Space					Output
R(t-9)	R(t-8)	R(t-7)	Q(t-2)	Q(t-1)	Q(t)
0.0000	0.0000	0.0000	0.0033	0.0033	0.0033
0.0000	0.0000	0.0000	0.0064	0.0064	0.0064
0.0000	0.0000	0.0000	0.0104	0.0104	0.0104

Table 4. Values of Correlation Coefficient Between Computed and Observed Flow With Respect to Progressive Addition of Hidden Neurons[a]

Added Neuron	Added Variable	R^2	Description Length of the Model
Linear	Q(t-2)	0.9265	1316
Linear	R(t-7)	0.9568	1225
Linear	R(t-9)	0.9735	1171

[a]The values of initial model with only three nonlinear hidden neurons are 0.8623 and 2345, respectively.

ranging from 300 to 600 m, is predominantly covered by deciduous forests. Soils are skeletal to shallow except near canals, where they are relatively deep. In this area the rocks are weathered and deep fissures are visible. The channel beds are rocky or graveled. The general response of this upper part of the basin to rains is quick. The lower one fifth of the basin consists of a flat-bottomed valley narrowing toward the outlet and having elevations ranging from 300 to 350 m. The area is predominantly cultivable and soils are deep and have flat slopes, and as such, the response of this area to rainfall is likely to be quite slow.

[32] For the bulk of the study, rainfall and runoff data on an hourly interval for Kolar basin during the monsoon season (July, August,and September) for three years (1987–1989) are used. The rainfall data available were in the form of areal average values in the basin. The total available data has been divided into two sets, calibration set (data during the years 1987–1988) and validation set (data during the year 1989). Different models for forecast lead times of up to 6 hours have been developed in the study. The parameters of the model are identified using the calibration data set, and the model is tested for its performance on the validation data set. The resulting hydrographs from the model are analyzed statistically using various evaluation measures.

3.2. Selection of Inputs to the Model

[33] One of the most important steps in the model development process is the determination of significant input variables. Usually, not all of the potential input variables will be equally informative since some may be correlated, noisy, or have no significant relationship with the output variable being modeled [*Maier and Dandy*, 2000]. Generally, some degree of a priori knowledge is used to specify the initial set of candidate inputs [e.g., *Campolo et al.*, 1999; *Thirumalaiah and Deo*, 2000]. Although a priori identification is widely used in many applications and is necessary to define a candidate set of inputs, it is dependent on an expert's knowledge and hence is very subjective and case-dependent. Intuitively, the preferred approach for determining appropriate inputs and lags of inputs involves a combination of a priori knowledge and analytical approaches [*Maier and Dandy*, 1997]. When the relationship to be modeled is not well understood, then an analytical technique, such as cross correlation, is often employed [e.g., *Sajikumar and Thandaveswara*, 1999; *Luk et al.*, 2000; *Silverman and Dracup*, 2000; *Coulibaly et al.*, 2000, 2001; *Sudheer et al.*, 2002]. The major disadvantage associated with using cross correlation is that it is only able to detect linear dependence between two variables. Therefore cross correlation is unable to capture any nonlinear dependence that may exist between the inputs and the output, and may possibly result in the omission of important inputs that are related to the output in a nonlinear fashion. *Bowden et al.* [2004], while reviewing the current state of input selection procedures in water resources applications, report that the cross-correlation methods represent the most popular analytical techniques for selecting appropriate inputs. It follows that there is good scope for addressing this issue in future studies.

[34] The current study employed a statistical approach suggested by *Sudheer et al.* [2002] to identify the appropriate input vector. The method is based on the heuristic that the potential influencing variables corresponding to different time lags can be identified through statistical analysis of the data series that uses cross correlations, autocorrelations, and partial autocorrelations between the variables in question. The cross correlation between the runoff and rainfall series at various lags (Figure 5) showed significant correlation at 7, 8, and 9 hours of rainfall lag on the flow at any time. The autocorrelation and partial autocorrelation function (Figure 5) suggests a significant correlation at 95% confidence level up to 2 hours of runoff lag. Note that this correlogram is for the stochastic

Table 3. Number of Data Points in Low-, Medium-, and High-Flow Categories (Validation Period)[a]

Category	Number of Points	Percentage of the Total Data
Low ($x < \mu$)	1783	81.97
Medium ($\mu \leq x \leq \mu + 2\sigma$)	369	16.97
High ($x > \mu + 2\sigma$)	23	1.06
Total	2175	100

[a]Here μ is the mean and σ is the standard deviation.

Table 5. Performance Statistics of LN Model at 1-Hour Lead Forecasting

Performance Index	Calibration	Validation
R	0.9735	0.9808
E	0.9478	0.9615
RMSE, m^3/s	47.5239	25.8935
Normalized RMSE	0.7346	0.8081
SEE	47.5896	25.9652
Normalized SEE	0.7356	0.8103

Table 6. Performance Indices of LN Models for Higher Forecasting Lead Time

	Forecast Lead Time, Hours					
Performance Index	2	3	4	5	6	12
			Calibration			
R	0.9459	0.9078	0.8662	0.8249	0.7824	0.5656
E	0.8947	0.8242	0.7503	0.6804	0.6121	0.3199
RMSE	67.4702	87.1991	103.9154	117.5587	129.5195	171.4949
Normalized RMSE	1.0428	1.3477	1.6059	1.8167	2.0014	2.6497
SEE	67.5634	87.3197	104.0591	117.7212	129.6985	171.7319
			Validation			
R	0.9557	0.9205	0.8739	0.8251	0.7814	0.4115
E	0.9119	0.8442	0.7621	0.6764	0.6093	0.1225
RMSE	39.2015	52.1573	64.4986	75.2776	82.7743	124.5567
Normalized RMSE	1.2227	1.6258	2.009	2.3429	2.5741	3.8516
SEE	39.3103	52.3021	64.678	75.4873	83.0052	124.907

component of the runoff series, i.e., after removing the deterministic components from the original series. Therefore a total number of five variables (three rainfall and two runoff) are identified as inputs according to *Sudheer et al.* [2002]. The input vector thus become [$R_{(t\text{-}9)}$ $R_{(t\text{-}8)}$ $R_{(t\text{-}7)}$ $Q_{(t\text{-}2)}$ $Q_{(t\text{-}1)}$], where R is the rainfall and Q is the river flow, and t indicate the time at which the forecast is required. The input and output variables are scaled between the range (0, 1), as sigmoid function is being used at the nonlinear hidden neuron.

3.3. Model Evaluation

[35] Quantitative assessments of the degree to which the model simulations match the observations are used to provide an evaluation of the model's predictive abilities. Frequently, evaluations of model performance utilize a number of statistics and techniques, usually referred to as "goodness-of-fit" statistics. Many of the principal measurements that are used in the hydrological literature have been critically reviewed by *Legates and McCabe* [1999]. Still, there is diversity in the use of global goodness-of-fit statistics to determine how well models forecast flood hydrographs. As a single evaluation measure is not available [*Sudheer and Jain*, 2003], a multicriteria assessment was performed in the current study with various goodness-of-fit statistics. These measures can be grouped into two types: relative and absolute. Relative goodness-of-fit measures are nondimensional indices, which provide a relative comparison of the performance of one model against another. In contrast, absolute goodness-of-fit statistics are measured in the units of the flow measurement. The criteria that are employed are the root-mean-square error (RMSE) between the observed and forecasted values, the coefficient of efficiency [*Nash and Sutcliffe*, 1970], the standard error of estimate (SEE), and the noise-to-signal ratio. The definitions of these evaluation criteria are provided in Table 1.

4. Results and Discussions

4.1. Model Development

[36] The input patterns were clustered using subtractive clustering algorithm [*Chiu*, 1994], which resulted in three clusters. The coordinates of the cluster center are presented in Table 2. The corresponding cluster radius was 0.0065. Note that the presented cluster coordinates are in the scaled domain (0, 1), as scaling of data is performed prior to ANN model building. It is observed that the rainfall information has no definite role in the clusters. It can be observed from Table 2 that the input space clusters fall in the low-flow region since a major portion (~82%) of the data set lies in this category. The percentage of total data falling to three different categories (based on summary statistics of the data) is presented in Table 3 for reference. Note that the final number of clusters was arrived at after various trials. The trials were conducted by varying the cluster radius, and the resulted clusters were subject to analysis of closeness. The analysis of closeness was performed by calculating the

Table 7. Observed and Predicted Statistical Properties of the River Flow Series

	Mean				Standard Deviation			
	Training		Validation		Training		Validation	
Lead Time, Hours	Observed	Computed	Observed	Computed	Observed	Computed	Observed	Computed
1	64.69	64.58	32.04	33.31	207.98	202.51	132.01	131.71
2	64.70	65.10	32.06	33.35	207.98	196.60	132.10	121.24
3	64.70	64.73	32.08	35.22	207.98	188.81	132.19	115.03
4	64.71	64.85	32.10	35.54	207.98	180.10	132.28	111.84
5	64.71	64.76	32.13	36.13	207.98	171.56	132.37	117.02
12	64.72	64.74	32.34	41.90	207.98	117.60	133.00	81.85

Table 8. Hydrograph Characteristics of LN Forecasting Models at Different Lead Times

		1 Hour	2 Hours	3 Hours	4 Hours	5 Hours	12 Hours
Percent error in volume	Training	0.1765	−0.6192	−0.0341	−0.2229	−0.0798	−0.0294
	Validation	−3.9592	−4.0162	−9.7813	−10.6981	−12.4582	−29.5560
Percent error in peak flow	Training	6.0963	2.6214	10.0509	11.3779	9.9084	58.5141
	Validation	2.2511	6.9256	15.3993	25.0632	21.4650	53.7734

Euclidian distance between any two cluster centers, and the final number of clusters was fixed such that the clusters do not overlap. Since the arrived number of clusters is only an indicative number (for initialization) of nonlinear hidden neurons, the performed analysis is reasonable.

[37] The above analysis suggested building ANN architecture with five inputs, three hidden neurons (nonlinear), and one output neuron. The weights of the hidden-output layer are computed using LSE. The step in the training algorithm to identify the basis variable that best fits to the current error (step 5 in the algorithm) suggested the variable $Q_{(t\text{-}2)}$. Accordingly, the initial model was modified by adding one linear hidden neuron, which connects to only $Q_{(t\text{-}2)}$ in the input vector. The error for the modified LN model is then computed. The performance of the model with respect to progressive addition of linear/nonlinear neuron is presented in Table 4.

[38] It can be observed from Table 4 that at each step a linear neuron gets added to the model, resulting in a final ANN architecture consisting of three nonlinear hidden neurons connected to all the input variables and three linear neurons connected to $Q_{(t\text{-}2)}$, $R_{(t\text{-}7)}$, and $R_{(t\text{-}9)}$, respectively. It is observed that the addition of the linear neuron $Q_{(t\text{-}2)}$ increased the explained variance by 6%, compared with the LN model that had only three nonlinear hidden neurons. This was followed by further increase in explained variance of 3%, and 2% by $R_{(t\text{-}7)}$ and $R_{(t\text{-}9)}$, respectively. The plausible reason for this improvement in explained variance is that the added linear neurons ($Q_{(t\text{-}2)}$, $R_{(t\text{-}7)}$, and $R_{(t\text{-}9)}$) provides information about the changing state of saturation of a basin, which according to *Campolo et al.* [1999] is vital in order to correctly predict the flow. Note that two of these linear neurons are lagged values of rainfall, which has a direct bearing on the interflow from the basin. This observation validates the earlier considerations for incorporating linear neurons in the hidden layer.

4.2. Model Performance

4.2.1. Generalization Properties of 1-Hour Lead Forecasts

[39] The values of the performance indices for the 1-hour lead forecast by the LN model for Kolar basin are presented in Table 5. The correlation statistic (R), which evaluates the linear correlation between the observed and computed runoff, is very good for the model during calibration as well as the validation period. The model efficiency (E) that evaluates the capability of the model in predicting runoff values away from the mean is found to be more than 94% during the calibration and more than 96% during the validation periods for the model, which according to *Shamseldin* [1997] is very satisfactory. The RMSE statistic, which indicates a quantitative measure of the model error in units of the variable, is good for the model, as is evidenced by the low values. Overall, for a 1-hour lead forecast, the performance of the LN model is very satisfactory.

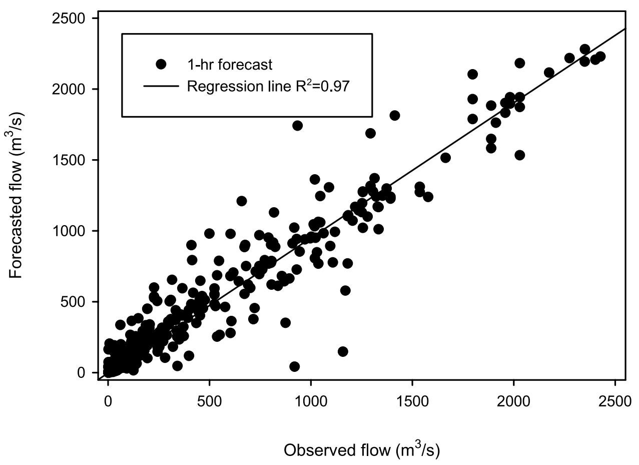
4.2.2. Forecasts at Higher Lead Time

[40] It is observed that most of the previous works based on ANN that are reported in the literature considered river flow forecasting at only one time step ahead (i.e., 1 hour or 1 day in advance) by providing information (input) to the model up to the current time step. However, as rightly pointed out by *Campolo et al.* [1999], a model that provides forecasts only at one time step ahead is impractical to use for flood warning or even water management. It may be noted that civil protection and water management agencies would greatly benefit from an increase in the prediction lead time. This would enable them to implement more efficient evacuation and protection plans for downstream areas in particular, as densely populated residential zones and areas characterized by presence of establishments of high economical value are exposed to the risks of flooding. These facts emphasize the need for improved flood forecasting techniques, aimed at extending and determining with higher accuracy the time of the occurrence of a flood farther downstream. Extending the prediction time horizon in the flood forecasting techniques would furthermore aid the planning of controlled release of water into appropriate storage volumes, thus mitigating potential damage.

[41] In the current study, the potential of LN model in forecasting the flows at higher lead times is also evaluated. This is achieved by training the LN model with $Q_{(t\text{-}1+i)}$ as the output, where i is the lead time of interest. LN models have been developed to forecast river flow up to 12 hours in advance ($i = 2, \ldots, 13$). The performance indices for these models are presented in Table 6. It is observed from Table 6 that the performance of the LN model deteriorates as the forecast lead time increases. However, the LN model is able to provide a reasonable forecast (with not less than 61% efficiency) of the river flow in Kolar basin up to 6 hours in

Table 9. Comparison of Model-Estimated Hydrograph Characteristics of One Typical Flood Event During Validation Period

	Event Peak: 2028.98 m^3/s		
Forecast Lead Time, Hours	Percent Error in Peak Flow, %	Estimated Peak Flow	Time Difference to Peak, Hours
1	−0.1642	2032.3110	0
2	9.1424	1843.4840	2
3	15.7808	1708.7910	1
4	22.1496	1579.5680	4
5	20.2751	1617.6010	0
12	53.7734	937.9286	6



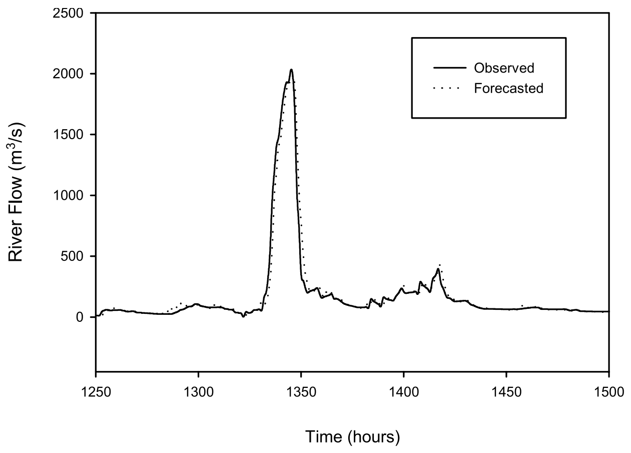


Figure 6. The observed and 1-hour-ahead forecasted river flows (a) during calibration and (b) for a typical validation flood event.

advance. The forecasts provided by the model at a lead time of 12 hours are not satisfactory.

4.2.3. Preservation of Summary Statistics

[42] The results from all the LN models are tested for their ability to preserve the statistical moments of the river flow series. The values of the first two moments, namely, mean and standard deviation, for the actual and forecasted computed river flow series, are presented in Table 7. From Table 7 it is apparent that all the models are able to preserve the first and second moments with reasonable accuracy. The standard deviation of the forecasted series gets deteriorated as the forecast horizon increases, and it is observed that the LN models for higher lead times are slightly over-predicting the flows.

4.2.4. Predicted Hydrograph Characteristics

[43] The performance indices discussed and considered so far provide relevant information on the overall performance of the models, but they do not provide specific information about the model performance at high flow, which is of critical importance in a flood forecasting context. Hence two additional storm specific evaluation measures are also considered: percentage error in predicted peak flow and percentage error in total runoff volume. Both these evaluation measures are computed as the ratio of deviation from observed value and the corresponding observed value expressed as percentage. These two indices are presented in Table 8 for all the LN models. The results indicate that the LN models slightly overestimate the peak flow at all lead times. However, the percent error is minimal at lower lead times (2.25% for 1-hour, and 6.9% for 2-hour forecasts). On the contrary, even though the total volume of the hydrograph is computed accurately by all the LN models during calibration, they underestimate the total discharge volume of the basin during validation. However, these errors are not very significant even at a lead time of 5 hours (12.45%).

[44] In order to critically examine the forecasting characteristics of the LN models, a typical flood event of peak flow (during the validation period) of magnitude 2028.98 m^3/s was considered, and the model predictions of this event were evaluated. The forecast characteristics such as error in peak flow estimation and time difference to peak are presented in Table 9. It is observed that as the lead time increases, the time to predicted peak flow also get lagged for the LN models. The results indicate that the LN models fail to identify the recession process accurately for larger lead time forecasts. One of the reasons for this appears to be related to the small number of training examples present in the subdomain corresponding to a higher flow range (see Table 2), from which a generalized nonlinear behavior of the process cannot be assessed. Note that the low- and medium-flow events are more influenced by local topography, soil moisture status, infiltration properties, and several other characteristics, often resulting in a complex rainfall-runoff relationship [*Zhang and Govindaraju*, 2000]. During the high-flow events, rainfall dominates the discharge at the stream, and other factors tend to have a minor role in the collective response. In order to assess the transformation of rainfall into runoff in such cases, more example data are hence required. It may also be noted that the models used spatially averaged rainfall information to forecast the flows. Figure 6 depicts the observed and forecasted flow by the LN model at a lead time of 1 hour. The forecasting potential of the LN model is clearly evident from Figure 6.

4.2.5. Predictive Uncertainty

[45] The predictive uncertainty of the models is evaluated by an index called noise-to-signal ratio. The unbiased SEE

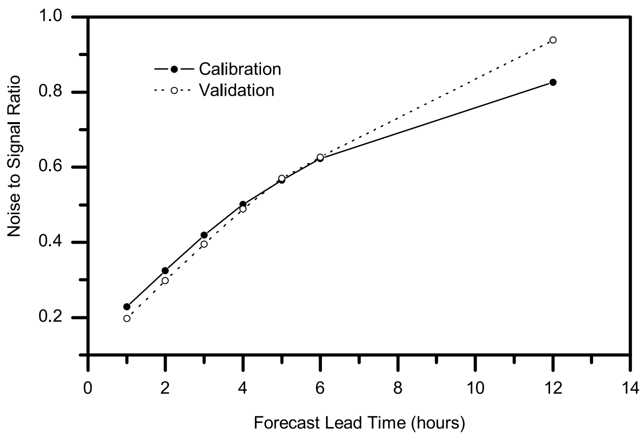


Figure 7. The noise-to-signal ratio for different LN models during calibration and validation.

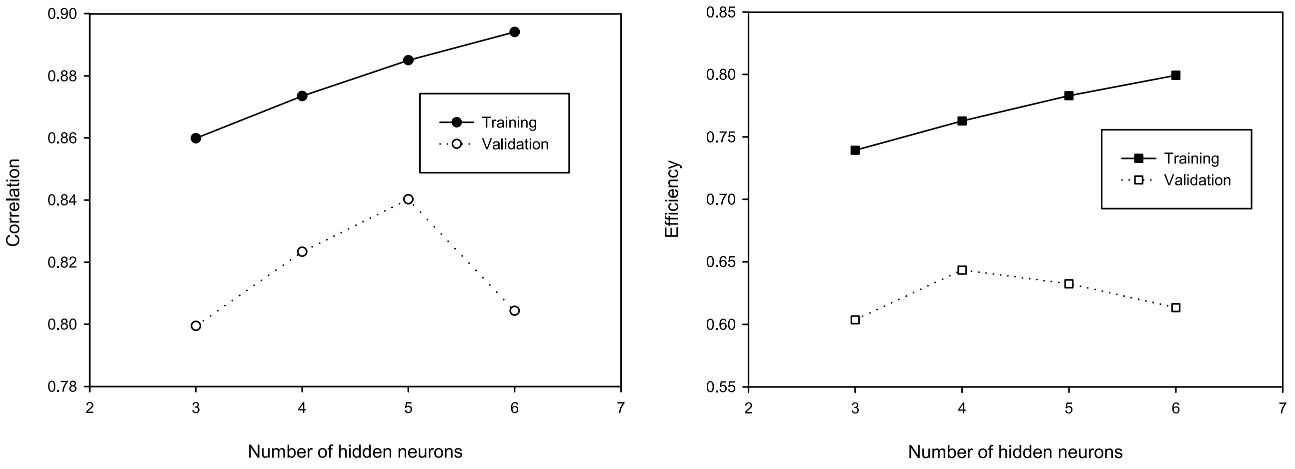


Figure 8. Variation of ANN model performance with number of nonlinear hidden neurons.

is a measure of the unexplained variance [*Tokar and Johnson*, 1999]. It is usually compared with the standard deviation of the observed values of the dependant variable (STD). The ratio of SEE to STD, called the noise-to-signal ratio, indicates the degree to which noise hides the information [*Gupta and Sorooshian*, 1985]. If the SEE is significantly smaller than STD, then the model can provide accurate predictions. On the contrary, if the ratio is greater than or equal to unity, then the model predictions will not be accurate [*McCuen*, 1993]. The noise-to-signal ratio of all the developed models is presented in Figure 7, which clearly depicts the adequacy of LN models. It may be noted that the noise-to-signal ratio of the LN model that forecasts 12 hours in advance is not greater than unity.

4.3. Comparison With Other Models

[46] In order to evaluate the LN model performance relative to other models, an independent ANN and a multiple linear regression (MLR) model have been developed for Kolar basin (1-hour lead forecasts). The number of hidden neurons in the ANN was varied from three to six, and it is observed that the ANN model achieves a highest efficiency of 84.02% during validation. The variations of performance indices (R and E) against the number of hidden neurons are shown in Figure 8, for both calibration and validation. It can be seen from Figure 8 that five hidden neurons are ideal for the ANN model. However, in order to arrive at this number, a large number of networks have been trained and tested for performance, which is not at all required in the proposed LN model. As expected, the performance of the MLR model was much inferior to ANN (an efficiency of 60.32% during calibration and 56.11% during validation). These results indicate that the LN model combines the strengths of both ANN and MLR models in a synergetic manner to achieve a better performance.

5. Remarks

[47] The foregoing discussions clearly illustrate the potential of the LN model in effectively modeling the rainfall-runoff process. The results indicate that providing information on the saturation level of the basin in terms of the lagged values of rainfall as linear hidden neurons helps effectively capture the linear portion of the flood hydrograph. An interesting observation is that the number of nonlinear hidden neurons considered initially, based on clustering of the input space, did not get increased during training of LN models for all forecast lead times. This observation is significant, as it may lead to development of a model architecture selection algorithm based on clustering of input space.

[48] It is worth mentioning that unlike other implementations of neural networks, the LN model includes linear terms explicitly in the model (equation (6)). This architecture and the proposed training algorithm have certain advantages, that if a process is linear by itself, the LN model will lead to a linear model (i.e., the final model would not contain any nonlinear hidden neuron) during the training process. The algorithm presented in this paper for training the LN model is easy to implement and does not involve any complex optimization procedure. The algorithm avoids overfitting by constraining the neurons in the hidden layer to minimize the description length of the model.

6. Summary and Conclusions

[49] This paper addresses the problem of forecasting the river flow on the basis of rainfall and runoff data. The paper mainly focused on three objectives: One was to demonstrate the formulation of a new modeling framework that uses a combination of linear and nonlinear neurons in the hidden layer of ANN. The second was to design a training algorithm for the proposed model, and the third was to evaluate the proposed model on a real-world case study. The concept of developing such a hybrid LN model stems from the studies that illustrate the role of a hidden layer in an ANN. While formulating the linear-neural (LN) model frame work, it was envisaged that in representing the rainfall runoff process, all the neurons in the hidden layer need not be nonlinear; instead some of them can be linear and they may represent the subprocess which shows a linear variation with time. The proposed LN model was developed and tested for its performance for forecasting river flow in Kolar basin, India. The analyses of the LN model developed for Kolar basin confirm the heuristic of incorporating linear

neuron in the hidden layer. It is observed that the linear neurons that are added in the final architecture of the LN model have a direct bearing on the interflow of the basin.

[50] The algorithm that has been designed for training the LN model was found to be simple to implement without the requirement of a complex optimization procedure. The algorithm removes the burden of selecting the best fit model among competing classes. A specific advantage of the algorithm is that it would result in a linear model if the process being modeled is purely linear. Performance evaluation of the LN model developed for Kolar basin suggests that the LN model is able to effectively capture the inherent nonlinearity in the rainfall-runoff process. The LN model is found to be able to forecast flows satisfactorily up to 6 hours in advance. A very close fit was obtained between computed and observed flows up to 1 hour in advance for the model, but forecasts at higher lead times (>6 hours) were found to be not effective. A comparison of the LN model performance with ANN and MLR models suggests that the LN model is able to combine the strengths of both these models effectively. The results of the study are highly encouraging and suggest that an LN modeling approach is viable for developing short-term forecasts of river flow series.

Appendix A: Description Length Computation

[51] If $\{y_i\}_{i=1}^{N}$ is a time series of N measurements and if $f(y_{i-1}, y_{i-2},......y_{i-d}; \Lambda_k)$ is a scalar function of d variables that is completely described by k parameters Λ_k, then the prediction error e_i is defined as [*Small and Tse*, 2002]

$$e_i = f(y_{i-1}, y_{i-2}, ...y_{i-d}; \Lambda_k) - y_i. \quad \text{(A1)}$$

It follows that Λ_k is the solution of

$$\min \sum_{i=1}^{N} e_i^2$$

for a fixed k. For any $\Lambda_k = (\lambda_1, \lambda_2, \ldots \lambda_k)$, the description length of the model $f(y_{i-1}, y_{i-2}, ...y_{i-d}; \Lambda_k)$ is given by the description length of the k parameters, computed as

$$M(k) = \sum_{j=1}^{k} \ln \frac{\gamma}{\delta_j}, \quad \text{(A2)}$$

where γ is the constant that gives the number of bits in the exponent of the floating point representation of λ_j. The precision δ_j of the optimal MDL model (for a fixed k) is computed based on the technique given by *Judd and Mees* [1995]. They showed that the optimal precision $(\delta_1, \delta_2, \ldots, \delta_k)$ is given by the solution of

$$\left(\mathbf{Q}^* \begin{bmatrix} \delta_1 \\ \delta_2 \\ : \\ : \\ \delta_k \end{bmatrix} \right) = \frac{1}{\delta_j}, \quad \text{(A3)}$$

where the matrix $\mathbf{Q}$ is obtained as the second derivative of the description length of the model errors $E(k)$ with respect to the model parameters Λ_k. It follows that

$$\mathbf{Q} = D_{\Lambda_k \Lambda_k} E(k), \quad \text{(A4)}$$

in which, $D_{\Lambda k \Lambda k}$ represents the second derivative. If errors are assumed to be Gaussian distributed with mean zero and a specific standard deviation σ, then

$$\mathbf{E(k)} = \frac{N}{2} + \ln\left(\frac{2\pi}{N}\right)^{N/2} + \ln\left(\sum_{i=1}^{N} e_i^2\right)^{N/2}. \quad \text{(A5)}$$

[52] Hence the description length of a model is calculated as follows.

[53] 1. Solving equation (A3) yields the precision with which one must specify each parameter.

[54] 2. From equations (A2) and (A5) the description length of the model $M(k)$ and model prediction errors $E(k)$ are obtained.

[55] **Acknowledgments.** We wish to express our sincere gratitude to two anonymous reviewers whose constructive suggestions and remarks that have greatly helped us to improve the quality of the manuscript. We also acknowledge the help rendered by P C Nayak, Scientist, National Institute of Hydrology, Roorkee, India.

References

Amorocho, J., and A. Brandstetter (1971), Determination of nonlinear functional response functions in rainfall-runoff processes, *Water Resour. Res.*, *7*(5), 1087–1101.

Bowden, G. J., G. C. Dandy, and H. R. Maier (2004), Input determination for neural network models in water resources applications: 1. Background and methodology, *J. Hydrol.*, *301*(1–4), 75–92.

Box, G. E. P., and G. M. Jenkins (1976), *Time Series Analysis, Forecasting and Control*, Holden-Day, Boca Raton, Fla.

Campolo, M., P. Andreussi, and A. Soldati (1999), River flood forecasting with a neural network model, *Water Resour. Res.*, *35*(4), 1191–1197.

Chiu, S. (1994), Fuzzy model identification based on cluster estimation, *J. Intell. Fuzzy Syst.*, *2*(3), 267–278.

Coulibaly, P., F. Anctil, and B. Bobee (2000), Daily reservoir inflow forecasting using artificial neural networks with stopped training approach, *J. Hydrol.*, *230*, 244–257.

Coulibaly, P., F. Anctil, and B. Bobee (2001), Multivariate reservoir inflow forecasting using temporal neural network, *J. Hydrol. Eng.*, *6*(5), 367–376.

Grayson, R. B., I. D. Moore, and T. A. McMahon (1992), Physically based hydrologic modeling: II. Is the concept realistic?, *Water Resour. Res.*, *28*(10), 2659–2666.

Gupta, V. K., and S. Sorooshian (1985), The relationship between data and the precision of parameter estimates of hydrologic models, *J. Hydrol.*, *81*, 57–77.

Hassibi, B., D. G. Stork, G. Wolff, and T. Watanbe (1994), Optimal brain surgeon: Extensions and performance comparisons, in *Advances in Neural Information Processing Systems*, vol. 6, edited by J. D. Cowan et al., pp. 263–270, Elsevier, New York.

Hsu, K., V. H. Gupta, and S. Sorooshian (1995), Artificial neural network modeling of the rainfall-runoff process, *Water Resour. Res.*, *31*(10), 2517–2530.

Ikeda, S., M. Ochiai, and Y. Sawaragi (1976), Sequential GMDH algorithm and its applications to river flow prediction, *IEEE Trans. Syst. Manage. Cybern.*, *6*(7), 473–479.

Jacoby, S. L. S. (1966), A mathematical model for non-linear hydrologic systems, *J. Geophys. Res.*, *71*(20), 4811–4824.

Jain, A., K. P. Sudheer, and S. Srinivasulu (2004), Identification of physical processes inherent in artificial neural network rainfall runoff models, *Hydrol. Processes*, *18*, 571–581.

Judd, K., and A. I. Mees (1995), On selecting models for nonlinear time series, *Physica D*, *82*, 426–444.

Karunanithi, N., W. J. Grenney, D. Whitley, and K. Bovee (1994), Neural networks for river flow prediction, *J. Comput. Civ. Eng.*, *8*(2), 201–220.

Kwok, T.-Y., and D.-Y. Yeung (1997), Constructive algorithms for structure learning in feed-forward neural networks for regression problems, *IEEE Trans. Neural Networks*, *8*(3), 630–645.

Le Cun, Y., J. S. Denker, and S. Solla (1990), Optimal brain damage, in *Advances in Neural Information Processing Systems*, vol. 2, edited by D. S. Touretzky, pp. 598–605, Elsevier, New York.

Legates, D.R., and G. J. McCabe Jr. (1999), Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation, *Water Resour. Res.*, *35*(1), 233–241.

Luk, K. C., J. E. Ball, and A. Sharma (2000), A study of optimal model lag and spatial inputs to artificial neural network for rainfall forecasting, *J. Hydrol.*, *227*, 56–65.

Maier, H. R., and G. C. Dandy (1997), Determining inputs for neural network models of multivariate time series, *Microcomput. Civ. Eng.*, *12*, 353–368.

Maier, H. R., and G. C. Dandy (2000), Neural networks for the prediction and forecasting of water resources variables: A review of modeling issues and application, *Environ. Model. Software*, *15*, 101–124.

McCuen, R. H. (1993), *Statistical Hydrology*, Prentice-Hall, Upper Saddle River, N. J.

Nash, J. E., and J. V. Sutcliffe (1970), River flow forecasting through conceptual models: 1. A discussion of principles, *J. Hydrol.*, *10*, 282–290.

Natale, I., and E. Todini (1976), A stable estimator for linear models: 2. Real-world hydrological applications, *Water Resour. Res.*, *12*(4), 672–676.

Reed, R. (1993), Pruning algorithms—A review, *IEEE Trans. Neural Networks*, *4*, 740–747.

Rissanen, J. (1978), Modeling by shortest data description, *Automatica*, *14*, 465–471.

Sajikumar, S., and B. S. Thandaveswara (1999), A non-linear rainfall-runoff model using an artificial neural network, *J. Hydrol.*, *216*, 32–55.

Sefton, C. E. M., and S. M. Howarth (1998), Relationships between dynamic response characteristics and physical descriptors of catchments in England and Wales, *J. Hydrol.*, *211*, 1–16.

Shamseldin, A. Y. (1997), Application of a neural network technique to rainfall runoff modeling, *J. Hydrol.*, *199*, 272–294.

Shamseldin, A. Y., A. E. Nasr, and K. M. O'Connor (2002), Comparison of different forms of the multi-layer feed-forward neural network method used for river flow forecasting, *Hydrol. Earth Syst. Sci.*, *6*, 671–684.

Silverman, D., and J. A. Dracup (2000), Artificial neural networks and long-range precipitation in California, *J. Appl. Meteorol.*, *31*(1), 57–66.

Small, M., and K. C. Tse (2002), Minimum description length neural networks for time series prediction, *Phys. Rev. E*, *66*(6), 1–12, doi:10.1103/PhysRevE.66.066701, 1–12.

Sudheer, K. P. (2000), Modeling hydrological processes using neural computing technique, Ph.D. thesis, Indian Inst. of Technol., Delhi, India.

Sudheer, K. P. (2005), Knowledge extraction from trained neural network river flow models, *J. Hydrol. Eng.*, *10*(4), 264–269.

Sudheer, K. P., and S. K. Jain (2003), Radial basis function neural networks for modeling stage discharge relationship, *J. Hydrol. Eng.*, *8*(3), 161–164.

Sudheer, K. P., and A. Jain (2004), Explaining the internal behaviour of artificial neural network river flow models, *Hydrol. Processes*, *18*(4), 833–844.

Sudheer, K. P., A. K. Gosain, and K. S. Ramasastri (2002), A data-driven algorithm for constructing artificial neural network rainfall-runoff models, *Hydrol. Processes*, *16*(6), 1325–1330.

Sudheer, K. P., P. C. Nayak, and K. S. Ramasastri (2003), Improving peak flow estimates in artificial neural network river flow models, *Hydrol. Processes*, *17*(3), 677–686.

Thirumalaiah, K., and M. C. Deo (2000), Hydrological forecasting using neural networks, *J. Hydrol. Eng.*, *5*(2), 180–189.

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

Wilby, R. L., R. J. Abrahart, and C. W. Dawson (2003), Detection of conceptual model rainfall runoff processes inside an artificial neural network, *Hydrol. Sci. J.*, *48*(2), 163–181.

Zhang, B., and R. S. Govindaraju (2000), Prediction of watershed runoff using Bayesian concepts and modular neural networks, *Water Resour. Res.*, *36*(3), 753–762.

M. Chetan and K. P. Sudheer, Department of Civil Engineering, Indian Institute of Technology Madras, Chennai, India 600 036. (chetan_iitm@yahoo.com; sudheer@iitm.ac.in)