Neural, Parallel, and Scientific Computations 17 (2009) 47-58

A CONNECTIONIST MODEL FOR RAINFALL PREDICTION

Bimal Dutta and Angshuman Ray
Institute of Engineering and Management
Salt Lake Electronics Complex, Calcutta 700 091, India
Srimanta Pal
Electronics and Communication Sciences Unit
Indian Statistical Institute, 203 B T Road, Calcutta 700 108, India
e-mail: srimanta@isical.ac.in
Dipak Chandra Patranabis
Department of Instrumentation and Electronics Engineering,
Jadavpur University, Salt Lake Campus, Kolkata – 700098, INDIA &
Heritage Institute of Technology, Kolkata – 700107
e-mail: dcp@iee.jusl.ac.in

Abstract: In this paper a neural network based method of local rainfall prediction is proposed. This method is developed based on past observations on various atmospheric parameters such as temperature, relative humidity, vapor presser, etc. We propose a neural network model whose architecture combines several multilayer perceptron networks (MLPs) to realize better performance after capturing the seasonality effect in the atmospheric data. We also demonstrate that the use of appropriate features can further improve the performance in prediction accuracy. These observations inspired us to use a feature selection MLP, FSMLP, (instead of MLP) which can select good features *on-line* while learning the prediction task. The FSMLP is used as a preprocessor to select good features. The combined use of FSMLP and SOFM-MLP results in a network system that uses only very few inputs but can produce good prediction.

Keywords – Rainfall, feature selection, multi-layer perceptron, neural networks, backpropagation, atmospheric science.

1. INTRODUCTION

There are many real life problems in which future events need to be predicted on the basis of past history. In such cases, knowledge of underlying laws governing the process can be very useful. The discovery of strong empirical regularities in observations on a given system can also help prediction. However, the laws underlying the behavior of a system are not easily discovered and the empirical regularities or periodicities are not always evident and can often be masked by noises.

Lower atmospheric parameters are used in various applications such as avionics, pollution dispersal, communication, etc. Therefore the accurate measurement or prediction of these parameters is necessary. Though perfect prediction of these parameters is hardly ever possible, neural networks can be used to obtain a reasonably good prediction in many cases (Tsintikidis, 1997; Salehfar, 1998; Pal et al., 2003; Sarma et al., 2005). Weather forecasting needs to estimate or predict atmospheric parameters (e.g., temperature, rainfall, relative humidity, wind speed, wind direction, atmospheric pressure, etc.) well in advance. Often it is very difficult to obtain an accurate prediction because of many other factors like topography of a place, surrounding structures and environmental pollution. The lower atmosphere is continuously changing. The accuracy of a forecasting system may be improved if the system considers these factors.

Rainfall is not a regular phenomena in all places. It has some seasonality effects. So the rainfall prediction problem is not similar as other regular atmospheric parameters like temperature, humidity, etc. Rainfall is also a time series data like atmospheric pressure, temperature, vapor pressure, relative humidity, radiation, etc. The rainfall pattern of Calcutta is shown in Figure 1. Some of the traditional time series analysis of the rainfall is shown in Table 2.

Here we focus on prediction of rainfall based on past measurements of various atmospheric parameters. Our assumption is that short-term changes in the dynamics of the atmosphere will be captured in the data available for forecasting. Normally, rainfalls are measured once in a day at different places using

Received February 10, 2009

rain gauge by the department of meteorology. Other parameters such as wind direction and its velocity, temperature, relative humidity, vapor pressure, radiation, etc. are also measured by the meteorologists.

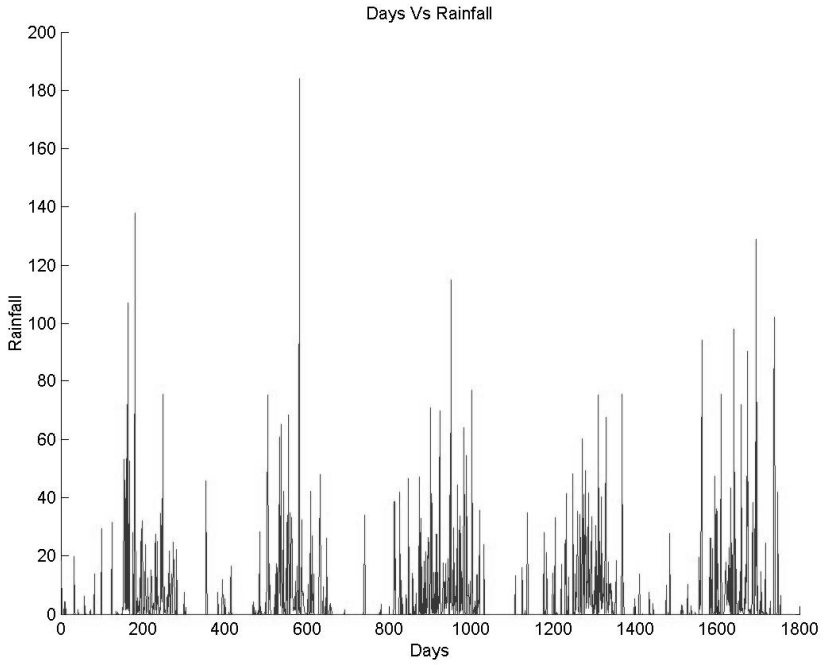

Figure 1. Actual rainfall at Dum Dum, Calcutta Station over consecutive 7 years (1989-95).

We have collected the following information for a day from the meteorology department: (1) mean sea level *pressure* at 1730 Hrs. and 0630 Hrs., (2) *vapor pressure* at 1730 Hrs. and 0630 Hrs., (3) *relative humidity* at 1730 Hrs. and 0630 Hrs., (4) *maximum temperature* at 1730 Hrs., (5) *minimum temperature* at 0630 Hrs., (6) *maximum radiation*, (7) *minimum radiation* and (8) *rainfall*. We have daily observation on these variables for the period 1989-95. Figure 1 shows the variation of the rainfall over the said period of 7 years.

In this paper first we study the effectiveness of multiplayer perceptron networks for prediction of rainfall. Next we design a hybrid network which uses both self-organizing feature map (SOFM) and MLP to realize a much better prediction system. Then we demonstrate that use of appropriate features cannot only reduce the number of features but also can improve the prediction accuracy. We then use a feature selection MLP, which can select good features *on-line* while learning the prediction task. This finally results in a network system that uses only very few inputs and can produce good prediction.

2. SOME POPULAR PREDICTION METHODS TO RAINFALL FORECASTING

In this section we consider two prediction model based on neural networks and statistical methods, and investigate their effectiveness in predicting the rainfall. We use the multi-layer perceptron (MLP) network and the auto-regressive (AR) model.

We make a brief, but comprehensive discussion of the MLP network because this is used later to design a more effective neural network for rainfall prediction.

The MLP network consists of several layers of neurons of which the first layer is known as the *input layer* and the last one is known as the *output layer*, remaining layers are called *hidden layers*. A typical MLP network is shown in Fig. 2. The nodes in successive layers are fully connected but there is no connection within a layer. Every node, in the hidden layers and the output layer computes the weighted sum of its inputs and apply a sigmoidal activation function to compute its output, which is then transmitted to the nodes of the next layer as input (Haykin, 2001). The main objective of the MLP learning is to set the connection weights in such a way that the error between the network output and the target output is minimized. During learning the network weights may be updated by several

methods of which the backpropagation technique is the most popular one. In this work we use the backpropagation learning.

It is known that under a fairly general assumption a single hidden layer is sufficient for a multilayer perceptron to compute an uniform approximation of a given training set (represented by the set of inputs and a desired (target) output) (Haykin, 2001). Hence in this study we restrict ourselves to a three-layer network, i.e., one hidden layer.

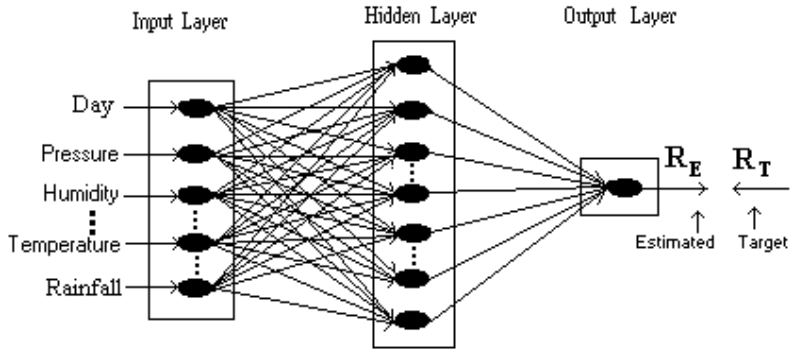

Figure 2. An MLP for rainfall forecasting.

In addition to MLP we also use *autoregressive* (AR) model. In AR model, rainfall at time t is predicted using a linear function of only past rainfall of a few days.

The auto-regressive (AR) model of order p, that is, AR(p) predicts the current (tth) value $x(t)$ of a variable x using its p previous observations such as $x(t-1)$, $x(t-2)$, …, $x(t-p)$. Mathematically, the AR(p) model can be expressed as $x(t) = a_0 + \Sigma_{1 \le i \le p} a_i x(t-i)$, where a_i's are the coefficients. For example, in rainfall prediction, the AR(p) model computes (predicts) the rainfall of the tth day (R(t)) based on last p days rainfall R(t–1), R(t–2), …, R(t–p).

We use an AR model of order three, i.e., the rainfall of the tth day is predicted based on the rainfall of past two days, i.e., days t–1, t–2. The performance of this AR model is shown in Table 2. In later section we shall discuss a feature selection mechanism that rejects rainfall information beyond past two days. In this regard, one can, of course, use model selection criteria such as Akaike-Information-Criteria (AIC) (Akaike, 1974).

3. DATA PREPARATION

MLP networks: For a particular day t, we have observation on 11 variables. Let us denote them by $\mathbf{x}(t) \in \Re^{11}$. Now let us assume that the rainfall for day t, i.e., R(t), are determined by the atmospheric conditions of past k days and $t-2$ date feature.

Thus we attempt to predict R(t) using $X(t) = (\mathbf{x}(t-1), \mathbf{x}(t-2), \ldots, \mathbf{x}(t-k)) + 1 \in \Re^{11k+1}$.
If $k = 2$, then we use $(\mathbf{x}(t-1), \mathbf{x}(t-2)) \in \Re^{23}$ to predict $Y_t = R(t)$.

We have **N** number of observations. Now we construct the input (X) and output (Y) data for training and test the network, where X= {X(N), X(N – 1), …, X(N – k)} and Y = {Y(N), Y(N – 1), …, Y(N – k)}. In order to train the network we use input-output pairs (X(i),Y(i)), i = N, N – 1, N – 2, …, N – k. In this case we have (N – k) pairs. Note that $Y_t = R(t)$. After obtaining (X, Y), we partition X (and also Y) randomly as X_{tr} (Y_{tr}) and X_{te} (Y_{te}) such that $X_{tr} \cup X_{te} = X$, $X_{tr} \cap X_{te} = \phi$. ($X_{tr}$, Y_{tr}) is then used for training the system and (X_{te}, Y_{te}) is used to test the system. So our MLP will have $11k$ +1 input nodes and 1 output nodes. In our data set N = 700 $|X_{tr}| = 600$, $|X_{te}| = 170$ and we use $k = 2$.

AR model: The rainfall, R(t) of the t-th day is determined using the rainfalls of three previous days, i.e., using R(t – 1), R(t – 2), R(t – 3).

3.1 Results

We have made several runs of the MLP net with different hidden nodes (n_h). Table 1 reports the average performance (average on 10 runs) on the test data for n_h = 5, 10, 15 and 20 nodes. Table 1 shows the cumulative percentage of prediction within different ranges. For example, the column with

n_h = 5 shows that on the test data the network could make prediction with $\leq \pm 2.5mm$ error in 83.67% cases. It is interesting to note that the networks with n_h= 5, 10, 15 and 20 perform reasonably well but the performance degrades with increase in the number of hidden nodes beyond 20.

Table 1: Cumulative percentage frequency for MLP Networks

Range	**% Frequency of rainfall for Test data**			
in mm	**nh=5**	**nh=10**	**nh=15**	**nh=20**
±0.5	44.89	51.02	51.02	53.06
±1.0	58.16	67.34	63.26	66.32
±1.5	75.51	78.57	76.53	77.55
±2.0	78.57	82.65	79.59	79.59
±2.5	83.67	85.71	83.67	83.67
±3.0	85.71	86.73	85.71	85.71
±3.5	87.75	87.75	87.75	87.75
±4.0	87.75	87.75	87.75	87.75
±4.5	87.75	89.79	88.77	88.77
±5.0	89.79	90.81	88.77	89.79
Max Dev	135.7	136.6	126.4	135.9
Avg Dev	13.3	8.2	3.3	6.3

Table 2 summarizes the performance of AR(3) model, MLP model and SOFM-MLP (self-organizing feature map MLP). The performance of the AR system exhibits a poorer performance than the two neural models.

The results obtained from MLP are satisfactory. But these results are not so good. One possible reason for this can be the presence of seasonality. This can be improved further. So we now propose a hybrid network which can account for seasonality of data. Our basic philosophy would be as follows. We group the data, X, into a set of homogeneous subgroups. Then for each subgroup we train a separate feed forward network. In this prediction, first we have to choose the appropriate trained MLP and then apply the test input to that net to get the prediction. The partitioning of the training data will be done using a self-organizing feature map (SOFM). Here we first briefly describe the SOFM, before describing the prediction network.

Table 2: Cumulative percentage frequency for SOFM-MLP, MLP, AR models

Cumulative Frequency for SOFM-MLP, MLP and AR models.			
Range	**SOFM-MLP**	**MLP**	**AR**
±0.5	87.73	83.53	4.95
±1.0	90.78	84.12	5.20
±1.5	93.28	85.88	5.24
±2.0	95.21	87.06	5.24
±2.5	95.77	88.23	5.24
±3.0	96.70	88.82	5.29
±3.5	97.08	88.82	5.36
±4.0	97.26	89.41	5.44
±4.5	97.64	90.00	5.52
±5.0	97.82	90.00	5.56

4. SOFM NETWORK

Kohonen's self organizing feature map (SOFM) has been successfully used in numerous applications such as pattern recognition (Kohonen, 1987), image processing (Nasarabadi, 1988), process control (Mark, 1988). Designing of classifiers (Mitra 1994) and other pattern recognition systems based on SOFM (Chi, 1995) are some of the most successful areas of its application. SOFM (Kohonen, 1990) has the interesting property of achieving a distribution of the weight vectors that approximates the

distribution of the input data. This property of the SOFM can be exploited to generate prototypes which in turn can partition the data into homogeneous groups. We want to use this property.

4.1 Architecture

The self-organizing feature map (SOFM) is basically a transformation $A^D_{SOFM} : \Re^p \rightarrow V(\Re^q)$ of higher dimensional (p) data to lower dimension (q) by an algorithm. This is often advocated for visualization of metric-topological relationships and distributional density properties of feature vectors (signals) $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ in $\Re^p$. In principle X can be transformed onto a display lattice in $\Re^q$ for any q. SOFM is implemented through a neural network as shown in Fig. 3. The visual display produced by A^D_{SOFM} helps to form hypothesis about topological structure present in X. Here we consider $(m \times n)$ structure and displays in $\Re^2$.

A SOFM network is shown in Fig. 3, where input vectors $\mathbf{x} \in \Re^p$ are distributed by a fan-out layer to each of the $(m \times n)$ output nodes in the competitive layer. Each node in this competitive layer has a weight vector $\mathbf{v}_{ij}$ attached to it where $i \in \{1,2, \ldots, m\}$ and $j \in \{1,2, \ldots, n\}$. This attached weight vector is equivalent to the prototype of the cluster. Let $O_p = \{\mathbf{v}_{ij}\} \subset \Re^p$ denotes the set of $m \times n$ weight vectors. O_p is (logically) connected to a display grid $O_2 \subset V(\Re^2)$. Now (i, j) in the index set $\{1,2, \ldots, m\} \times \{1,2, \ldots, n\}$ is the logical address of the cell. There is a one-to-one correspondence between the p dimensional vector $\mathbf{v}_{ij}$ and the (i, j)th cell, i.e., $O_p \leftarrow O_2$ where where $i \in \{1,2, \ldots, m\}$ and $j \in \{1,2, \ldots, n\}$.

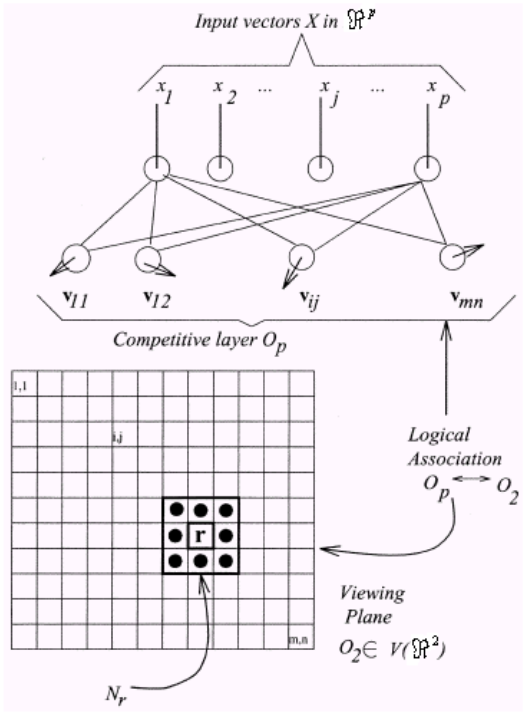

Figure 3. The SOFM network architecture.

4.2 Algorithm

The feature mapping is an iterative algorithm. It starts with (usually) a random initialization of the weight vectors $\mathbf{v}_{ij}$ for $i = 1,2, \ldots, m$ and $j = 1,2, \ldots, n$. For notational clarity we suppress the double subscripts in $\mathbf{v}$. Now let an input vector $\mathbf{x} \in \Re^p$ enter the network and let s denote the current iteration number. During sth iteration, find a vector $\mathbf{v}$ which is obtained after $(s - 1)$th iteration, i.e., $\mathbf{v}_{r,s-1}$, that best matches the input vector $\mathbf{x}$ in the sense of minimum Euclidean distance in $\Re^p$. This vector, $\mathbf{v}_{r,s-1}$ has an (logical) ``image" which is the cell in O_2 with subscript r (as shown in Fig. 4). Next a topological (spatial) neighborhood $N_r(s)$ centered at r is defined in O_2, and its display cell neighbors are located. A 3×3 window, $N(r)$, centered at r corresponds to updating nine prototypes in $\Re^p$. Finally, $\mathbf{v}_{r,s-1}$ and other weight vectors associated with cells in the spatial neighborhood $N_s(r)$ are updated using the rule

$$\mathbf{V}_{k,s} = \mathbf{v}_{k,s-1} + H_{rk}(s)(\mathbf{x} - \mathbf{v}_{k,s-1}).$$

Here r is the index of the ``winner" prototype

$$r = \arg\, min_k \{|\mathbf{x} - \mathbf{v}_{k,s-1}|\}$$

and $|*|$ is the Euclidean norm on $\Re^p$.

The function $H_{rk}(s)$ which expresses the strength of interaction between cells r and k in O_2 usually decreases with s, and for a fixed s it decreases as the distance (in O_2) from cell r to cell k increases. $H_{rk}(s)$ is usually expressed as the product of a learning parameter α_s and a lateral feedback function $g_s(dist(r,k))$. A common choice for g_s is $g_s(dist(r,k)) = exp\{-dist^2(r,k)/\sigma_s^2\}$. α_s and σ_s both decrease with s. The topological neighborhood $N_r(s)$ also decreases with s. This scheme, when repeated long enough, usually preserves spatial order in the sense that weight vectors which are metrically close in $\Re^p$ generally have, at termination of the learning procedure, visually close images in the viewing plane. We repeat the SOFM for $(500 \times m \times n)$ steps (Kohonen, 1990).

5. SOFM-MLP HYBRID NETWORK

The architecture of this hybrid network is shown in Fig. 4. It has eight layers. The first layer with p nodes scales the data -- it is the scaling interface between user and the system at the input side. The second and third layers constitute the SOFM layer. The output of the scaling layer is fed as input to the SOFM layer. So the second layer has p nodes. There are complete connections between layers 2 and 3 as discussed earlier for the SOFM net.

Let the number of nodes in the output layer of the SOFM network is K. So, there are K MLP networks, each of which receives p inputs. Consequently, the fourth layer has Kp nodes. These Kp nodes constitute the input layer of a set of K MLP networks. Without loss of generality, we assume that each of the K MLP networks has only one hidden layer, although it could be more than one and also it can vary for different MLP nets. Let the nodes in layer four be numbered as N_i, i=1, 2, …, Kp. Nodes N_1 to N_p will be the input nodes of the first MLP (M_1); nodes N_{p+1} to N_{2p} will be input nodes of the second MLP (M_2); Similarly, nodes $N_{(K-1)p+1}$ to N_{np} will be the input nodes of Kth MLP, M_K. As mentioned earlier, $p = 11k+1$.

The j-th input node of MLP M_i gets two inputs from the previous layers: the jth normalized input (say x_j) and the output of the ith node of the SOFM network (say o_i). Each node of layer four computes the product of the two inputs it receives, i.e., the jth input node of M_i computes x_j. o_i as output and passes it on the first hidden layer of the MLP network, M_i. Since only one output of the SOFM layer will be one and rest of the outputs will be zero, only one of the MLP networks, which is connected to the winner node, will get the normalized input unattenuated. While all inputs to each of the remaining (K-1) MLP will be zero. So only one of the MLP will be activated.

Since we assume only one hidden layer, the nodes in layer six are the output nodes of the MLP nets. Each MLP, M_i will have two output nodes. Let us denote these nodes by O^6_i where the index i corresponds to the ith MLP, M_i. Layers 4 - 6 together constitute the MLP layer in Fig. 4. The outputs of this MLP-layer are then aggregated in layer seven which has just one node. Let us denote this node by R. Now nodes O^6_i, $\forall$ i=1,2, …, K are connected to node m and O^6_{i2}, $\forall$ i=1,2, …, K are connected to node M. All connection weights between layers 6 and 7 are set to unity and node R computes the weighted sum of all inputs as the output which is then passed to the scaling layer. Note that, the network architecture ensures that the aggregated output that is fed to the scaling layer is nothing but the output of the MLP corresponding to the winning node of the SOFM network.

The prototypes generated by SOFM not only preserves topology but also density. We want to have this density preservation property. If there is a dense cluster in the input space SOFM will place more prototypes there, consequently we will have more competitive MLPs for dense regions. Hence finer details of the process can be modeled better resulting in enhancement of overall performance. None of clustering algorithm has this density matching property, thereby making the self-organizing map somewhat superior to clustering algorithm.

5.1 Training the SOFM-MLP Hybrid Network

First X_{tr} is normalized by the input normalization (i.e., scaling) layer. Then with the normalized X_{tr} the SOFM net is trained. Once the SOFM training is over, then X_{tr} is partitioned into K subsets, $X^{(l)}_{tr}$, l = 1, 2, ..., K as follows:

$$X^{(l)}_{tr} = \mathbf{x}_i \in \Re^p \mid \| \mathbf{x}_i - \mathbf{v}_l \| = min_j \| \mathbf{x}_i - \mathbf{v}_j \| .$$

In other words, $X^{(l)}_{tr}$ is the set of input vectors for which the l-th prototype, $\mathbf{v}_l$ of the SOFM becomes the winner. Let $Y^{(l)}_{tr}$ be the set of output vectors associated with vectors in $X^{(l)}_{tr}$. Now we train K multilayer perceptron nets $M_1, M_2, ..., M_K$, where M_l is trained with $X^{(l)}_{tr}$, $Y^{(l)}_{tr}$. Note that, each of M_l, $l = 1, 2, ..., K$ will have the same number of nodes in the input layer, i.e., $p = 11k$ and the same number of nodes in the output layer. But the number of nodes in the hidden layer for different M_l could be different. This training is done off-line and during training, we do not consider the output of the SOFM. In fact, we do not feed the input to SOFM for training the MLP.

Once the training of both SOFM and K MLP's is over, we are in a position to use the hybrid net for prediction of rainfall.

Consider the input vector $\mathbf{x}(t) \in \Re^{11k}$ (this will be generated based on 11 observations on each of the past k days). Now $\mathbf{x}(t)$ is applied to the first layer. The first layer normalizes it and the normalized input then goes to the SOFM layer. $\mathbf{x}(t)$ makes the output of only one of the K SOFM output nodes, say of the lth node, high (1) and sets the rest (K–1) outputs to zero. The normalized $\mathbf{x}$(t) and output of the ith SOFM node are now fed to the ith MLP M_i, $i = 1, 2, ..., K$. Consequently, only the lth MLP will be active and rest of the MLPs will be inactive. The integrated output from the MLP layer will be nothing but the output of the lth MLP, which will then be scaled back to the original scale by the output scaling layer -- and we get the prediction for the rainfall of day $t + 1$.

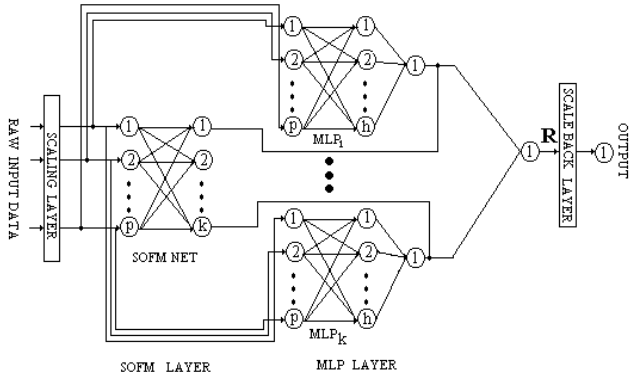

Figure 4. A hybrid neural net for rainfall prediction.

Table 3: Cumulative percentage frequency table for SOFM-MLP when observations on past 2 days are used as input (Test sample)

Range	**% Frequency of rainfall for test data**			
in mm	**n_h=5**	**n_h=10**	**n_h=15**	**n_h=20**
±0.5	82.3	81.7	85.3	83.0
±1.0	84.3	83.7	86.5	85.0
±1.5	87.0	85.5	88.0	87.5
±2.0	87.8	87.5	88.5	88.2
±2.5	88.8	89.0	89.3	89.2
±3.0	89.2	89.7	90.3	89.5
±3.5	89.5	90.0	90.7	89.8
±4.0	89.8	90.3	90.7	90.0
±4.5	90.5	90.8	90.8	90.5
±5.0	90.8	91.0	91.3	91.2
Max Dev	77.2	76.8	79.0	75.9
Avg Dev	1.9	1.9	1.6	1.8

5.2 Results

Table 3 depicts the performance of the SOFM-MLP network on the test data when each of the K (= 8) MLPs uses n_h= 10, n_h = 15 and n_h = 20 nodes in the hidden layer. For the SOFM layer we have used 8 nodes thereby the training data were partitioned into 8 homogeneous subgroups. For this data set the choice of 8 was made based on a few experiments. In this case, use of more than 8 nodes results in some clusters with very few data points.

Each MLP is trained 10 times with different random initialization and Table 3 represents the average prediction accuracy over these runs. From Table 2 we find that within an error of ±1.0 *mm*, the SOFM-MLP shows an improvement between 83.7% to 86.5% over the direct use of MLP. In case of an error of ±5.0 *mm*, this improvement is about 90.8% to 91.3%. If we consider the maximum deviation and the average deviation, we also find consistently better results for SOFM-MLP.

6. ONLINE FEATURE SELECTION AND HYBRID NETWORK

We have observed two things: the hybrid network works better than MLP and the choice of good features improves the prediction accuracy. Therefore, if we can do *online* feature selection, i.e., select the good features while learning the prediction task, we can probably further improve the performance of the network and this can also tell us about various important features responsible for rainfall. This may help us to get a better insight into the rainfall process. We try to do these now.

6.1 *Online* Feature Selection Technique

Here we choose good features that will improve the rainfall prediction. An *online feature selection* network selects the good features while learning the estimation task. We can probably further improve the performance of our network and this can also tell us about various important features responsible for this prediction.

In a standard multilayer perceptron network, the effect of some features (inputs) can be eliminated by not allowing them into the network. The "partially useful" features can be identified and attenuated according to their relative usefulness (Pal and Chantalapudi 1997, Pal et al. 2003, Sarma et al 2005). This can be realized by associating an adaptive gate to each input node. The gate should be modeled in such a manner that for a good feature, it is completely opened and the feature is passed unattenuated into the net; while for a bad feature, the gate should be closed tightly. On the other hand for a partially important feature, the gate could be partially open. Mathematically, the gate is modeled by a function F with a tunable parameter. The degree to which the gate is opened determines the goodness of the feature. We multiply an input feature value by its attenuation value and the modulated feature value is passed into the network. The gate functions attenuate the features before they propagate through the net so we may call these gate functions as *attenuator functions*. A simple way of identifying useful gate functions is to use *s*-type (or sigmoidal) functions with a tunable parameter which can be learnt using training data.

Let $F: R \rightarrow [0,1]$ be an attenuation function associated with each of the p input nodes. If x is the node input then $xF(\gamma)$ is the node output. Thus, $xF(\gamma_i)$ can be viewed as the activation function of the ith input layer node, where γ_i is a parameter (not a connection weight) of the activation function. Thus, the input layer nodes act as "neurons" (i.e., have internal calculations). Notice that $F(\gamma_i)$ acts as a *fixed* multiplier for *all* input values of the ith feature once γ_i is known. The function F can have various forms. In the experiments described below we use the attenuation function

$$F(\gamma) = \frac{1}{1+e^{-\gamma}}$$

Thus, the ith input node attenuates x_i by an amount $F(\gamma_i) \in (0,1)$, where the input "weight" γ_i is a parameter to be learnt during training. If $F(\gamma_i)$ is close to 0, we may choose to eliminate input feature x_i: this is how the FSMLP accomplishes feature selection. How do we learn γ_i? If we regard the input layer of the FSMLP with nodes as in Fig. 2, it can be viewed as the "first"' hidden layer in a standard MLP. Then the back-propagation formulae for the MLP are simply extended backwards into this new first layer to adjust the p γ_i s during training. Let

q = number of nodes in the first hidden (not input) layer;
μ = learning rate for the parameters of the attenuator membership functions,
η = learning rate for the connection weights,
$w_{ij}^{ih}(t)$ = weight connecting ith node of the input layer to the jth node of the first hidden layer for the tth iteration and
δ_j^1 = error term for the jth node of the first hidden layer.
$F'(\gamma_i)$ = derivative of F at γ_i,
$F: R \rightarrow (0,1)$ = attenuator function with argument γ_i for input node i.

Table 4: Feature attunations after 5000 iterations

Feature	Exp1	Exp2	Exp3	Exp4	Exp5
date	0.994408	0.995326	0.994349	0.994401	0.994349
pmin(t-2)	0.037687	0.034179	0.036586	0.036168	0.036586
pmax(t-2)	0.049841	0.050345	0.050567	0.050500	0.050567
vpmax(t-2)	0.571023	0.567013	0.610024	0.606581	0.610024
vpmin(t-2)	0.035260	0.041261	0.038821	0.039218	0.038821
rhmin(t-2)	0.047426	0.047426	0.047426	0.047426	0.047426
rhmax(t-2)	0.981611	0.974963	0.981208	0.980619	0.981208
tmax(t-2)	0.047426	0.047426	0.047426	0.047426	0.047426
tmin(t-2)	0.047426	0.047426	0.047426	0.047426	0.047426
rain(t-2)	0.047426	0.047426	0.047426	0.047426	0.047426
rmax(t-2)	0.047426	0.047426	0.047426	0.047426	0.047426
rmin(t-2)	0.047426	0.047426	0.047426	0.047426	0.047426
pmin(t-1)	0.647053	0.656292	0.690670	0.688052	0.690670
pmax(t-1)	0.014436	0.025190	0.014467	0.015558	0.014467
vpmax(t-1)	0.980603	0.962653	0.983048	0.982541	0.983048
vpmin(t-1)	0.129533	0.081264	0.099814	0.101237	0.099814
rhmin(t-1)	0.993324	0.964764	0.992428	0.991945	0.992428
rhmax(t-1)	0.005520	0.007016	0.005383	0.005442	0.005383
tmax(t-1)	0.066341	0.060743	0.057793	0.057599	0.057793
tmin(t-1)	0.978657	0.971962	0.959644	0.955657	0.959644
rain(t-1)	0.001737	0.002357	0.001768	0.001802	0.001768
rmax(t-1)	0.057392	0.067241	0.062358	0.064471	0.062358
rmin(t-1)	0.986055	0.978115	0.987440	0.987164	0.987440

Table 5: Feature Selection by Voting Scheme Using Table 4

Feature	Exp1	Exp2	Exp3	Exp4	Exp5	Frequency	Decision
date	Selected	Selected	Selected	Selected	Selected	5	Selected
pmin(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
pmax(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
vpmax(t-2)	Rejected	Rejected	Selected	Selected	Selected	5	Selected
vpmin(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rhmin(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rhmax(t-2)	Selected	Selected	Selected	Selected	Selected	5	Selected
tmax(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
tmin(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rain(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rmax(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rmin(t-2)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
pmin(t-1)	Selected	Selected	Selected	Selected	Selected	5	Selected
pmax(t-1)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
vpmax(t-1)	Selected	Selected	Selected	Selected	Selected	5	Selected
vpmin(t-1)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rhmin(t-1)	Selected	Selected	Selected	Selected	Selected	5	Selected
rhmax(t-1)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
tmax(t-1)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
tmin(t-1)	Selected	Selected	Selected	Selected	Selected	5	Selected
rain(t-1)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rmax(t-1)	Rejected	Rejected	Rejected	Rejected	Rejected	0	Rejected
rmin(t-1)	Selected	Selected	Selected	Selected	Selected	5	Selected

It can be easily shown that the learning rule for connection weights remains the same for all layers except for $w_{ij}^{ih}(t)$. The update rule for $w_{ij}^{ih}(t)$ and γ_i are:

$$w_{ij}^{ih}(t) = w_{ij}^{ih}(t-1) - \eta x_i \delta_j^1 F(\gamma_i(t-1))$$

$$\gamma_i(t) = \gamma_i(t-1) + \mu x_i \left(\sum_{j=1}^{q} w_{ij}^{ho} \delta_j^1 F'(\gamma_i(t-1)) \right).$$

The p weights γ_i are initialized with values that make $F(\gamma_i) = \frac{1}{1+e^{-\gamma_i}}$ close to 0 for all i. Consequently, $x_i F(\gamma_i)$ is small at the beginning of training, so the FSMLP *allows only a very small "fraction" of each input feature value to pass into the standard part of the MLP*. As the network trains, it selectively allows only important features to be active by increasing their attenuator weights (and hence, increasing the multipliers of x_i associated with these weights) as dictated by the gradient descent. The training can be stopped when the network has learnt satisfactorily, i.e., the mean squared error is low or the number of iteration reaches a maximum limit. Features with *low* attenuator weights are eliminated from the feature set.

In this scheme we only consider those features whose attenuation weight values at the end of the run are ≥ 0.5 considering 0.04 or less as initialized values. Now we apply this feature selection method on our rainfall data and it selects only 8 features out of 23 features (Table 5). The FS MLP technique is trained to the whole data set so as to choose the important features. The FS technique chooses a total of 8 features out of the 23 features as can be inferred from the result of Table 5. As Table 5 reveals that FSMLP rejects the following features pmin(t – 2), pmax(t – 2), vpmin(t – 2), rhmin(t – 2), tmax(t – 2), tmin(t – 2), rain(t – 2), rmax(t – 2), rmin(t – 2), pmax(t – 1), vpmin(t – 1), rhmax(t – 1), tmax(t – 1), rain(t – 1), rmax(t – 1). It is an interesting observation that the today's rainfall does not depend on the previous days, rainfalls. But the network does not reject the following features: date, vpmax(t – 2), rhmax(t – 2), pmin(t – 1), vpmax(t – 1), rhmin(t – 1), tmin(t – 1), rmin(t – 1). Date feature indicates the seasonality effect of the place.

6.2 Results

In order to select the good features, we train the FSMLP using the entire data set. And after the features are selected, we train the SOFM-MLP with the selected set of features. Table 4 displays the attenuation factors of the 23 features after training the FSMLP.

For each network, the training is stopped when the prediction error on the validation set started increasing. We have made 10 experiments each with MLP and SOFM-MLP. Interestingly, in all cases but two, the training error and validation error exhibited identical behavior.

Table 6: Cumulative percentage frequency table for MLP using selected features.

Range	**% Frequency of rainfall for Test data**			
in mm	**nh=5**	**nh=10**	**nh=15**	**nh=20**
±0.5	83.5	82.4	83.5	83.5
±1.0	83.5	83.5	84.1	84.1
±1.5	84.7	84.7	85.9	84.1
±2.0	86.5	87.1	87.1	85.3
±2.5	86.5	87.1	88.2	85.9
±3.0	88.2	88.2	88.8	87.6
±3.5	89.4	89.4	88.8	87.6
±4.0	91.2	90.6	89.4	89.4
±4.5	91.8	91.2	90.0	91.2
±5.0	92.4	92.4	90.0	91.8
Max Dev	125.6	125.7	125.9	126.4
Avg Dev	3.3	3.2	3.2	3.3

Table 7: Cumulative percentage frequency table for SOFM-MLP using selected features

Range in mm	% frequency of rainfall for test data			
	$n_h = 5$	$n_h = 10$	$n_h = 15$	$n_h = 20$
±0.5	82.62	90.22	87.73	90.04
±1.0	85.45	92.09	90.78	92.46
±1.5	87.704	93.46	93.28	93.28
±2.0	92.15	94.28	95.21	94.65
±2.5	93.85	95.03	95.77	94.84
±3.0	94.92	95.77	96.71	95.40
±3.5	96.24	95.96	97.08	95.40
±4.0	97.20	96.33	97.26	95.77
±4.5	97.90	96.52	97.64	95.96
±5.0	98.08	96.71	97.83	96.71
Max Dev	72.6	64.4	62.1	69.7
Avg Dev	1.6	1.1	1.1	1.4

Tables 6 and 7 depict the average performance of MLP and SOFM-MLP using the selected features in conjunction with a validation data. Since in these cases we have used only 8 input features, we have restricted the maximum number of nodes in the hidden layer to 20 only.

Comparing Table 7 with Table 3 we find that in this case too there is a marginal improvement in performance for SOFM-MLP with the selected features. A comparison of Table 7 with Table 6 clearly shows that again SOFM-MLP outperforms the conventional MLP. The most important point is that we can use only a few features to get good results.

7. CONCLUSION AND DISCUSSION

From what have been presented above on concludes that:

(1) The proposed hybrid SOFM-MLP network consistently performs better than the conventional MLP network.
(2) Use of gradient as features instead of the raw observations can reduce the required size of the network and make the training task simpler yet achieving better performance.
(3) Feature selection is an important factor for better prediction of atmospheric parameters. In this regards our FSMLP turns out to be an excellent tool that can select good features while learning the prediction task.
(4) The combined use of FSMLP and SOFM-MLP results in an excellent paradigm for prediction of atmospheric parameters.

There are couple of other areas where we need to do experiments. For example, we plan to use FSMLP and SOFM-MLP set for prediction of other atmospheric parameters. We would also like to investigate their usefulness in different pattern recognition problems.

REFERENCES

1. Tsintikidis D., Haferman J.L., Anagnostou E.N., Krajewski W.F., & Smith T.F. (1997). A neural network approach to estimating rainfall from spaceborne microwave data. *IEEE GeoScience and Remote Sensing*, v. 35, pp. 1079-1093.
2. Salehfar H., & Benson S.A. (1998). Electric utility coal quality analysis using artificial neural network techniques. *Neurocomputing*, v. 23, pp. 195-206.
3. Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, v. AC-19, pp. 716-723.
4. Kohonen, T., Torkkola, K., Shozakai, M., Kangas, J., & Venta, O. (1987). Microprocessor implementation of a large vocabulary speech recognizer and phonetic typewriter for Finnish and Japanese. *Proc. European Conference of Speech Technology*, (Edinberg, 1987), pp. 377-380.
5. Nasarabadi, N. M., & Feng, Y. (1988). Vector Quantization of images based on Kohonen self-organizing feature maps. *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88*, (San Diego, Cal., 1988), pp. I-101-I-108.
6. Marks, K. M., & Goser, K. F. (1988). Analysis of VLSI process data based on self-organizing feature maps. *Proc. Neuro-Nimes'88*, (Nimes, France), pp. 337-347.

7. Mitra, S., & Pal, S. K. (1994). Self-organizing neural network as a fuzzy classifier. *IEEE Trans. Syst. Man, Cyberc*., v. 24, pp. 385-398.
8. Chi, Z., Wu, J., & Yan, H. (1995). Handwritten numeral character recognition using self-organizing maps and fuzzy rules. *Pattern Recognition*, v. 28, pp. 59-66.
9. Kohonen, T. (1990). The self-organizing map. *Proc. IEEE*, v. 78, pp. 1464-1480.
10. Pal, N. R., & Chintalapudi, K. (1997). A connecionist system for feature selection. *Neural, Parallel & Scientific Computation*, v. 5, pp. 359-381.
11. Haykin, S. (2001). *Neural Networks A Comprehensive Foundation*. Second Edition, Pearson Education, Singapore.
12. Pal, N. R., Pal, S., Das J., & Majumdar, K. (2003). SOFM-MLP: A hybrid neural network for atmospheric temperature prediction. *IEEE. Trans. Geosci. Remote Sens*., v. 41, pp. 2783-2791.
13. Sarma, D.K., Konwar, M., Das, J., Pal, S., & Sharma, S. (2005). A soft computing approach for rainfall retrieval from the TRMM microwave imager. *IEEE Trans. on Geoscience and Remote Sensing*, v. 43, pp. 2879-2885.