SOFM–MLP: A Hybrid Neural Network for Atmospheric Temperature Prediction

Nikhil R. Pal, *Senior Member, IEEE*, Srimanta Pal, Jyotirmoy Das, and Kausik Majumdar

***Abstract*—Here, first we study the effectiveness of multilayer perceptron networks (MLPs) for prediction of the maximum and the minimum temperatures based on past observations on various atmospheric parameters. To capture the seasonality of atmospheric data, with a view to improving the prediction accuracy, we then propose a novel neural architecture that combines a self-organizing feature map (SOFM) and MLPs to realize a hybrid network named SOFM–MLP with better performance. We also demonstrate that the use of appropriate features such as temperature gradient can not only reduce the number of features drastically, but also can improve the prediction accuracy. These observations inspired us to use a feature selection MLP (FSMLP) instead of MLP, which can select good features online while learning the prediction task. 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.**

***Index Terms*—Atmospheric science, backpropagation, feature selection, neural networks, self-organizing feature map (SOFM), temperature forecasting.**

I. Introduction

THE MEASUREMENT and prediction of lower atmospheric parameters are necessary for various kinds of applications such as avionics, pollution dispersal, and communication. Though perfect prediction is hardly ever possible, neural networks can be used to obtain a reasonably good prediction in many cases [1], [2]. Weather forecasting requires estimation of temperature, rainfall, humidity, wind speed, wind direction, atmospheric pressure, etc., well in advance. Often, it is very difficult to get an accurate prediction result because of many other factors like topography of a place, surrounding structures, and environmental pollution. The accuracy of a forecasting system may be improved if it can account for all these factors.

Here, we focus on prediction of temperature based on past measurements of various atmospheric parameters, and our assumption is that short-term changes in the dynamics will be captured in the data available for forecasting. Normally, temperatures are measured twice a day at different heights and places using radiosonde (i.e., balloon floating) techniques. Other parameters, such as wind direction and its velocity, are also measured by the meteorologists.

We have collected the following information for a day from the meteorology department: 1) mean sea level *pressure* at 1730 and 0630 Hrs; 2) vapor pressure at 1730 and 0630 Hrs; 3) relative humidity at 1730 and 0630 Hrs; 4) maximum temperature at 1730 Hrs; 5) minimum temperature at 0630 Hrs; and 6) rainfall. We have daily observation on these variables for the period 1989 to 1995.

In this paper, first we describe our proposed hybrid network, which combines a self-organizing feature map (SOFM) and a multilayer perceptron network (MLP) to realize a much better prediction system. Then, we demonstrate that the use of appropriate features can not only reduce the number of features, but also can improve the prediction accuracy. Motivated by the results with computed features, we then use a feature selection MLP, which can select good features online while learning the prediction task. This, finally, results in a network system that uses only very few inputs and can produce good prediction.

II. Some Popular Prediction Methods and Their Application to Temperature Forecasting

In this section we consider four prediction methods and investigate their effectiveness in predicting the maximum and the minimum temperature. We use the MLP and radial basis function (RBF) networks and the autoregressive (AR) and linear regressive (LR) models.

The MLP network consists of several layers of neurons of which the first one is the input layer, and the last one is the output layer, remaining layers are called hidden layers. There are complete connections between the nodes in successive layers but there is no connection within a layer. Every node, except the input layer nodes, computes the weighted sum of its inputs and apply a sigmoidal function to compute its output, which is then transmitted to the nodes of the next layer [3]. The objective of MLP learning is to set the connection weights such that the error between the network output and the target output is minimized.

In addition to MLP, we also use the radial basis function network [3], autoregrassive model and the linear regression model. In AR model, temperature at time t is predicted using a linear function of only past temperature of a few days; while the LR model predicts temperature as a linear function of observations on other parameters including temperature.

The RBF network is a three layered network: input layer, basis function layer (hidden layer as in MLP) and output layer. Each node in the hidden layer represents a basis function. We use Gaussian basis functions for all nodes. Here, the input layer and the hidden layer are fully connected. Each output node uses a linear activation function; in other words, each

output node computes the weighted sum of its inputs. Let $\mathbf{W}^k_{\mathrm{RBF}}$ be the vector of connection weights between the input nodes and the kth RBF node. Then the output of the kth RBF node is $h^k_{\mathrm{RBF}} = \exp(-\|\mathbf{i} - \mathbf{W}^k_{\mathrm{RBF}}\|^2/\sigma_k^2)$, where σ_k is the spread of kth RBF function and $\mathbf{i}$ is the input vector. The output of the jth output node o^j_{RBF} is then computed as $o^j_{\mathrm{RBF}} = \sum_{k=1}^{n_h} w_c^{kj} h^k_{\mathrm{RBF}}$, where w_c^{kj} is the connection weight between kth RBF node and jth output node.

We use the MATLAB neural network tool box to realize the RBF net. This training algorithm for RBF does not use any unsupervised processing such as clustering of data. The network starts with one RBF node using one of training data points as the center of the Gaussian function. Then it finds the data point with the highest error, which is used as the center of a new RBF node. The connection weights between hidden layer and the output layer are then adjusted minimizing the square error. The process is continued till either the error goal in terms of sum of square errors (SSE) is achieved or the number of RBF nodes attains a given maximum value. In our implementation, the maximum number of nodes was set to 50, and the error goal on SSE was 0.01.

The autoregressive model of order p, i.e., $AR(p)$, predicts the current (tth) value $x(t)$ of a variable x using its p previous observations such as $x(t-1), x(t-2), \ldots, x(t-p)$. Mathematically, the $AR(p)$ model can be written as $x(t) \quad a_0 - \sum_{i=1}^{p} a_i x(t-i)$, where a_i's are the coefficients. For example, in temperature prediction, the $AR(p)$ model computes (predicts) the maximum temperature of the tth day [$T^{\max}(t)$] based on last p days maximum temperatures $T^{\max}(t \quad 1), T^{\max}(t \quad 2), \ldots, T^{\max}(t \quad p)$.

A linear regression model can be used to predict the temperature of the tth day as a linear function of several other variables. The LR model can be described as $y(t) = a_0 + \sum_{i=1}^{q} a_i x_i(t-1)$, where a_i's are the coefficients. For example, in temperature prediction, the LR model computes the tth day's maximum (or minimum) temperature $T^{\max}(t)$ [or $T^{\min}(t)$] based on previous days observation on maximum temperature [$x_1 \quad T^{\max}(t-1)$], minimum temperature [$x_2 = T^{\min}(t-1)$], vapor pressure [$x_3 = V(t-1)$], relative humidity [$x_4 = H(t \quad 1)$], rain fall [$x_5 = R(t \quad 1)$], $\cdots$, pressure ($x_q \quad P(t-1)$). Here, q is the number of predictor variables used in the LR model.

We use an AR model of order three, i.e., the maximum (or minimum) temperature of the tth day is predicted based on the maximum (or minimum) temperature of past three days, i.e., days $t-1, t-2, t-3$. The order three has been chosen based on experiments. Also, in a later section, we shall see that a feature selection mechanism rejects temperature information beyond past three days. In this regard, one can, of course, use model selection criteria such as Akaike information criteria (AIC) [4].

A. Data Preparation

MLP and RBF Nets: For a particular day t, we have observation on nine variables. Let us denote them by $\mathbf{x}(t) \in R^9$. Now, let us assume that the maximum and the minimum temperatures for day t, i.e., $T^{\max}(t)$ and $T^{\min}(t)$, are determined by the atmospheric conditions of past k days.

TABLE I
CUMULATIVE PERCENTAGE FREQUENCY FOR MLP NETWORKS

Range in °C	% Frequency of Temperature for test data							
	$n_h = 10$		$n_h = 15$		$n_h = 20$		$n_h = 25$	
	max	min	max	min	max	min	max	min
±0.5	26.2	27.1	29.3	29.6	27.2	28.5	27.0	27.8
±1.0	52.5	53.7	54.7	57.1	53.1	55.7	52.2	53.9
±1.5	70.5	70.2	71.4	73.4	71.3	72.1	69.3	70.4
±2.0	81.2	81.9	83.6	85.2	81.2	83.5	80.5	82.0
±2.5	87.3	88.5	88.2	89.7	87.6	89.1	86.7	87.2
±3.0	90.5	93.6	91.7	92.7	91.5	92.1	90.8	91.0
Max Dev	6.3	5.9	6.3	5.8	6.2	5.7	7.0	6.6
Avg Dev	1.2	1.2	1.1	1.1	1.1	1.1	1.2	1.2

Thus, we attempt to predict $(T^{\max}(t), T^{\min}(t))$ using $X(t) = (\mathbf{x}(t-1), \mathbf{x}(t-2), \ldots, \mathbf{x}(t-k)) \in R^{9k}$. If $k = 2$, then we use $(\mathbf{x}(t-1), \mathbf{x}(t-2)) \in R^{18}$ to predict $Y_t = (T^{\max}(t), T^{\min}(t))$.

Let N be the total number of days for which observations are available. So, we can construct the input (X) and output (Y) data for training where $X = \{X(N), X(N-1), \ldots, X(N-k)\}$ and $Y = \{Y(N), Y(N-1), \ldots, Y(N-k)\}$. In order to train the network, we use input–output pairs $(X(i), Y(i)), i = N, N-1, N-2, \ldots, N-k$. In this case, we have $N-k$ pairs. Note that $Y_t = (T^{\max}(t), T^{\min}(t))$. After obtaining (X, Y), we partition X (and also Y) randomly as $X_{\mathrm{tr}}(Y_{\mathrm{tr}})$ and $X_{\mathrm{tc}}(Y_{\mathrm{tc}})$ such that $X_{\mathrm{tr}} \bigcup X_{\mathrm{te}} = X$, $X_{\mathrm{tr}} \bigcap X_{\mathrm{te}} = \phi$. $(X_{\mathrm{tr}}, Y_{\mathrm{tr}})$ is then used for training the system and $(X_{\mathrm{tc}}, Y_{\mathrm{tc}})$ is used to test the system. So, our MLP will have $9k$ input nodes and two output nodes. In our dataset, $N = 2555$, $|X_{\mathrm{tr}}| = 2285$, $|X_{\mathrm{tc}}| = 270$, and we use $k = 3$.

AR Model: The maximum (or minimum) temperature $T^{\max}(t)$ [or $T^{\min}(t)$] of the tth day is determined using the maximum (or minimum) temperatures of three previous days, i.e., using $T^{\max}(t \quad 1)$, $T^{\max}(t \quad 2)$, and $T^{\max}(t \quad 3)$ [or $T^{\min}(t-1), T^{\min}(t-2)$, and $T^{\min}(t-3)$].

LR Model: The maximum (or minimum) temperature $T^{\max}(t)$ [or $T^{\min}(t)$] of the tth day is determined using previous days' observations on the maximum temperature, minimum temperature, vapor pressure, relative humidity, change of temperature, change of vapor pressure, change of relative humidity, and rainfall.

B. Results

We have made several runs of the MLP net with different hidden nodes. Table I reports the average performance (average on ten runs) on the test data with number of hidden nodes, $n_h = 10$, 15, 20, and 25. It shows the cumulative percentage of prediction within different ranges. For example, the fourth row of the "max" column with $n_h = 10$ shows that on the test data the network could make prediction with $\leq \pm 2$ °C error in 81.2% cases. Similarly, the net with $n_h = 10$ can predict the minimum

TABLE II
CUMULATIVE PERCENTAGE FREQUENCY TABLE FOR RBF, AR, AND LR MODELS

Range in °C	% Frequency of Temperature for test data					
	RBF		AR model		LR model	
	max	min	max	min	max	min
±0.5	29.45	28.23	27.03	28.83	27.85	28.23
±1.0	53.05	53.40	55.26	52.80	54.24	51.61
±1.5	70.98	68.95	71.37	66.13	70.16	67.34
±2.0	80.65	81.66	80.44	78.23	80.47	78.05
±2.5	87.52	88.92	87.90	85.27	87.01	84.52
±3.0	91.34	91.15	91.34	90.72	91.13	90.15
Max. Dev.	8.374	5.906	7.080	6.291	8.923	6.801
Avg. Dev.	1.161	1.173	1.162	1.201	1.171	1.218

temperature with ± 2 °C error in 81.9% cases on the test data. It is interesting to note that the networks with $n_h = 10, 15$, and 20 perform reasonably well, but the performance degrades with increase in the number of hidden nodes beyond 20. Within ± 2 °C error, the best result that the networks could achieve for the maximum temperature is about 84%.

Table II summarizes the performance of the RBF network and the AR(3) and LR models. The RBF network (with 50 RBF nodes) is found to produce a little worse results than the MLP. The performance of the AR and LR systems is quite comparable, and they exhibit a little poorer performance than the two neural models.

Although the MLP and RBF results are satisfactory, they are not very good. One possible reason for this can be the presence of seasonality. So, we now propose a hybrid network which can account for seasonality of data. Our basic philosophy would be as follows. We group the vectors in (X) into a set of homogeneous subgroups. Then for each subgroup we train a separate feedforward network. For 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 as well as the selection of the desired trained MLP will be done using a self-organizing feature map. So, before describing the prediction network, we first briefly describe the SOFM.

III. SOFM NETWORK

Kohonen's self-organizing feature map has been successfully used in numerous applications [5]–[8]. SOFM [9] has the interesting property of achieving a distribution of 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.

A. Architecture

The self-organizing feature map is an algorithmic transformation $A^{D}_{\mathrm{SOFM}} : R^p \to V(R^q)$ that 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 R^p. SOFM is implemented through a neural architecture as shown in Fig. 1, and it is believed to be similar in some ways to the biological neural network. The visual display produced by SOFM can be used to form hypotheses about topological structure present in X. We concentrate on $(m \times n)$ displays in R^2, but in principle X can be transformed onto a display lattice in R^q for any q.

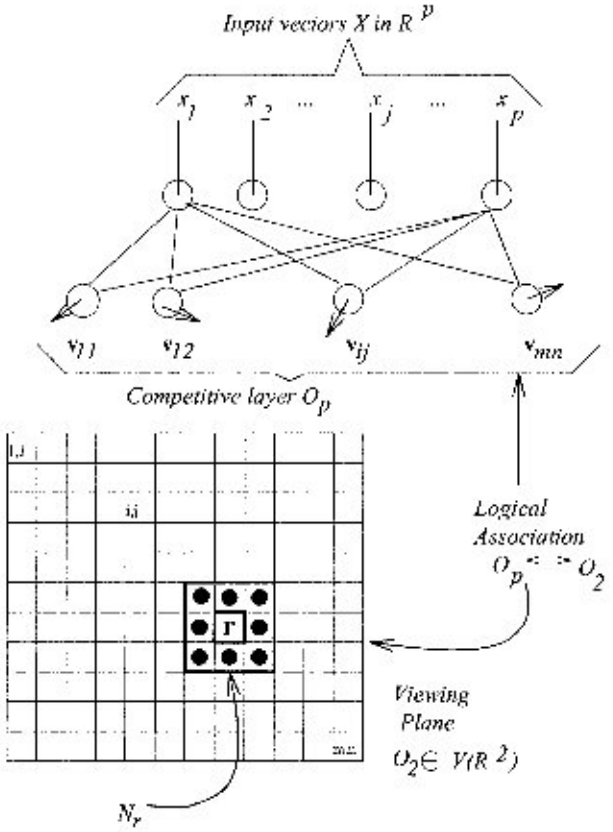

Fig. 1. SOFM network architecture.

As shown in Fig. 1, input vectors $\mathbf{x} \in R^p$ are distributed by a fan-out layer to each of the $(m \times n)$ output nodes in the competitive layer. Each node in this layer has a weight vector (prototype) $\mathbf{v}_{ij}$ attached to it. Let $O_p = \{\mathbf{v}_{ij}\} \subset R^p$ denote the set of $m \times n$ weight vectors. O_p is (logically) connected to a display grid $O_2 \subset V(R^2)$. (i,j) in the index set $\{1,2,\ldots,m\} \times \{1,2,\ldots,n\}$ is the logical address of a cell. There is a one-to-one correspondence between the $m \times n$ p-vectors $\mathbf{v}_{ij}$ and the $m \times n$ cells $(\{i,j\})$, i.e., $O_p \approx O_2$.

The feature mapping algorithm starts with a random initialization of the weight vectors $\mathbf{v}_{ij}$. For notational clarity, we suppress the double subscripts. Now, let $\mathbf{x} \subset R^p$ enter the network, and let s denote the current iteration number. Find $\mathbf{v}_{r,s-1}$ that best matches $\mathbf{x}$ in the sense of minimum Euclidean distance in R^p. This vector has a (logical) "image" that is the cell in O_2 with subscript r. 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 R^p. Finally, $\mathbf{v}_{r,s-1}$ and other weight

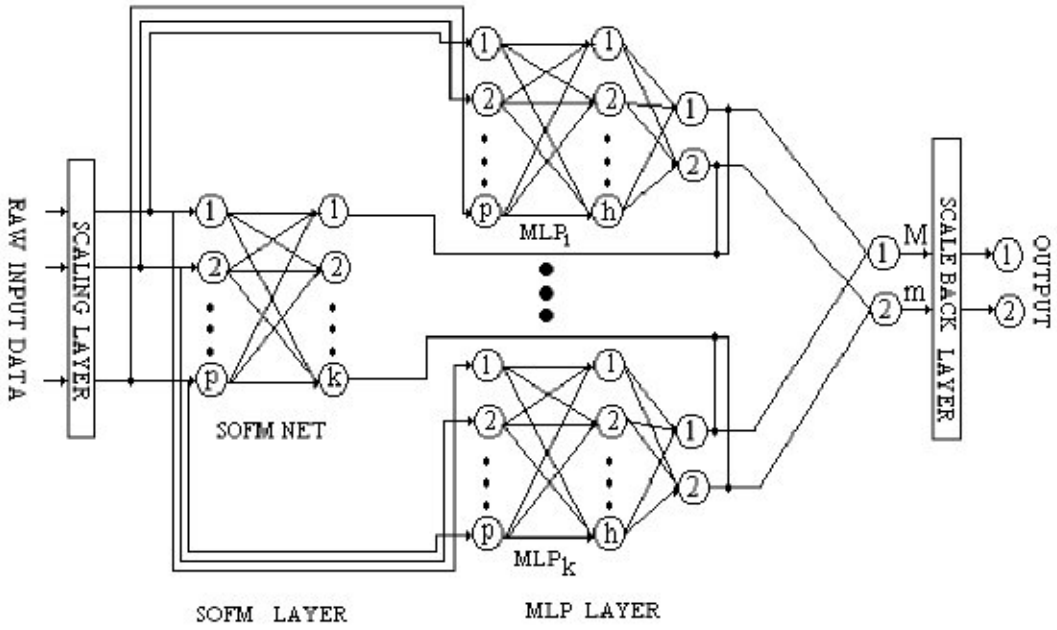

Fig. 2. Hybrid neural net for temperature forecasting.

vectors associated with cells in the spatial neighborhood $N_s(r)$ are updated using the rule

$$\mathbf{v}_{i,s} = \mathbf{v}_{i,s-1} + H_{ri}(s)(\mathbf{x} - \mathbf{v}_{i,s-1}). \quad (1)$$

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

$$r = \underbrace{\arg\min}_{i}\{\|\mathbf{x} - \mathbf{v}_{i,s-1}\|\} \quad (2)$$

and $\|*\|$ is the Euclidean norm on R^p. The function $H_{ri}(s)$ which expresses the strength of interaction between cells r and i 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 i increases. $H_{ri}(s)$ is usually expressed as the product of a learning parameter α_s and a lateral feedback function $g_s(\text{dist}(r,i))$. A common choice for g_s is $g_s(\text{dist}(r,i)) = c^{-\text{dist}^2(r,i)/\sigma_s^2}$. 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 R^p have visually close images in the viewing plane. We repeat the SOFM for $(500 \times m \times n)$ steps [9].

IV. SOFM–MLP Hybrid Network

The architecture of this hybrid network is shown in Fig. 2. 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 net be 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 it can vary for different MLP nets. Let the nodes in layer four be numbered as $N_i, i = 1, 2, \ldots, Kp$. 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_{Kp} will be the input nodes of Kth MLP, M_K. As mentioned earlier, $p = 9k$.

The jth input node of MLP M_i gets the jth normalized input (say, x_j) and passes it on to the first hidden layer of M_i. The output of the ith node of the SOFM (say, α_i) is connected to the output of every node of the last layer of M_i. The product of the MLP output and the SOFM output then moves to layer 7. The product can be computed using an additional layer with two neurons for each MLP. Since only one of the SOFM outputs will be one, and the rest will be zero, only one of the MLPs will pass its output unattenuated to layer 7. The remaining $(k-1)$ MLPs will transfer zero to layer 7.

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_{ij}^6 where the index i corresponds to the ith MLP, M_i, and $j = 1, 2$, where 1 corresponds to the minimum temperature and 2 corresponds to the maximum temperature. Layers 4–6 together constitute the MLP layer in Fig. 2. The outputs of this MLP-layer are then aggregated in layer seven which has just two nodes one for the minimum temperature and the other for the maximum temperature. Let us denote these two nodes by m and M. Now nodes $O_{i1}^6, \forall\, i = 1, 2, \ldots, K$ are connected to node m and O_{i2}^6, $\forall\, i = 1, 2, \ldots, K$ are connected to node M. All connection weights between layers 6 and 7 are set to unity and nodes m and M compute the weighted sum of all inputs as the output which are 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 net.

At this point, readers might wonder why are we using a self-organizing map, why not a clustering algorithm! The prototypes

generated by SOFM not only preserve topology but also density. We want to exploit this density preservation property. Because of the density matching property of SOFM, if a particular region of the input space contains frequently occurring stimuli, it will be represented by a larger area in the feature map than a region of the input space where the stimuli occur less frequently [3, p. 460]. Consequently, if there is a dense area in the input space SOFM will place more prototypes there. So, we will have more competitive MLPs for dense regions. Hence finer details of the process can be modeled better and this will result in an enhancement of the overall performance. k-means type clustering algorithms do not have such a density matching property.

A. Training the SOFM–MLP Hybrid Net

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, X_{tr} is partitioned into K subsets, $X_{\mathrm{tr}}^{(l)}$, $l = 1, 2, \ldots, K$ as follows:

$$X_{\mathrm{tr}}^{(l)} = \{\mathbf{x}_i \in R^p \mid \|\mathbf{x}_i - \mathbf{v}_l\| = \min_j \|\mathbf{x}_i - \mathbf{v}_j\|\}.$$

In other words, $X_{\mathrm{tr}}^{(l)}$ is the set of input vectors for which the lth prototype, $\mathbf{v}_l$ of the SOFM becomes the winner. Let $Y_{\mathrm{tr}}^{(l)}$ be the set of output vectors associated with vectors in $X_{\mathrm{tr}}^{(l)}$. Now we train K multilayer perceptron nets $M_1, M_2, \ldots, M_K$, where M_l is trained with $(X_{\mathrm{tr}}^{(l)}, Y_{\mathrm{tr}}^{(l)})$. Note that each of M_l, $l = 1, 2, \ldots, K$ will have the same number of nodes in the input layer, i.e., $p = 9k$ and the same number of nodes in the output layer, i.e., 2. But the number of nodes in the hidden layer for different M_l could be different. This training is done offline 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 MLPs is over, we are in a position to use the hybrid net for prediction of temperatures.

Suppose an input vector $\mathbf{x}(t) \subset R^{9k}$ comes (this will be generated based on nine 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, $i = 1, 2, \ldots, K$. 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 maximum and the minimum temperatures of day t 1.

B. Results

Table III 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 eight nodes, and thereby the training data were partitioned into eight homogeneous subgroups. For this dataset, the choice of eight was made based on a few experiments. In this case, use of more than eight nodes results in some clusters with very few data points. Each MLP is trained ten times with different random initialization, and Table III represents the average prediction accuracy over these runs. Comparing Table III with Table I, we find that within ± 1 °C error, the SOFM–MLP shows an improvement between 2.7% to 7.8% over the direct use of MLP. This improvement is reduced to about 2.2% to 7.4% within +2 °C error. If we consider the maximum deviation and the average deviation, we also find consistently better results for SOFM–MLP. Comparing SOFM–MLP (Table III) with another local predictor, the RBF network (Table II), we find that all three architectures of SOFM–MLP significantly outperform the RBF net for all deviations less than equal to ± 2° C.

TABLE III
CUMULATIVE PERCENTAGE FREQUENCY TABLE FOR SOFM–MLP WHEN OBSERVATIONS FROM THE PAST THREE DAYS ARE USED AS INPUT

Range in °C	% Frequency of temperature for test data					
	n_h = 10		n_h = 15		n_h = 20	
	max	min	max	min	max	min
±0.5	32.1	31.2	33.2	31.1	32.0	30.6
±1.0	60.3	59.3	61.2	59.6	60.7	58.8
±1.5	76.2	76.3	76.7	75.1	76.1	74.8
+2.0	88.6	87.3	88.0	87.4	87.8	86.1
+2.5	91.8	91.6	92.1	91.3	91.8	90.1
+3.0	95.1	93.7	94.7	93.2	93.1	93.2
Max. Dev.	5.74	6.03	5.80	5.81	5.61	5.62
Avg. Dev.	1.06	1.07	1.06	1.08	1.07	1.09

V. PREDICTION WITH COMPUTED FEATURES

So far, we have used the entire information available for the past three days to predict the temperatures for the next day. As a result, the number of input features becomes 27, making the learning task a difficult one.

For any learning task, the use of appropriate features is a key factor toward determining the success of the learning process. In this case, also, if we can use some derived features that are better suited for the task at hand, then we may expect to get better prediction. With a view to achieving this, we have used local gradients of the temperature sequences as features. The local gradient is computed as follows. Suppose $T^{\max}(t-4)$, $T^{\max}(t-3)$, $T^{\max}(t-2)$, and $T^{\max}(t-1)$ are the maximum temperatures recorded for the last four days. Then, the temperature gradients or changes in temperature are $T^{\max}(t-4) - T^{\max}(t-3)$, $T^{\max}(t-3) - T^{\max}(t-2)$, and $T^{\max}(t-2) - T^{\max}(t-1)$. Similarly, three such components can be computed for the minimum temperature. Here, we use 15 features containing nine features giving the atmospheric conditions of today (day t) and six temperature gradients as discussed above. The advantage with this scheme is that: 1) it reduces the number of input features, and 2) it gives an idea to the MLP network about the changes in the maximum and the minimum temperatures. This can make the learning task simpler. Our results in the next section reveal that it is indeed the case.

A. Results

Table IV shows the performance of the MLP and SOFM–MLP on the test data with the new features discussed above. Comparing Table IV with Table I, we find that with a smaller architecture (small number of input units), the performance of the ordinary MLP is consistently better. For deviations less than or equal to $\pm 2\,^{\circ}$C, the performance of the MLP network with gradients as features is consistently better than the corresponding MLP using 27 features. This is clearly a significant improvement because with the new features we are using a much smaller network.

Comparing columns of SOFM–MLP in Table IV with Table III, we find that for deviations less than or equal to $\pm 1.5\,^{\circ}$C, SOFM–MLP using gradient as features exhibits consistently better performance than the corresponding SOFM–MLP using all 27 features. For deviations less than or equal to $\pm 2\,^{\circ}$C, the performance of SOFM–MLP more or less remains the same. Table IV also reveals that the maximum deviation and average deviation are better for SOFM–MLP than direct use of MLP.

TABLE IV
CUMULATIVE PERCENTAGE FREQUENCY TABLE FOR MLP AND SOFM–MLP USING TEMPERATURE GRADIENTS AS FEATURES

Range in °C	% Frequency of Temperature for test data							
	MLP				SOFM-MLP			
	$n_h = 10$		$n_h = 15$		$n_h = 10$		$n_h = 15$	
	max	min	max	min	max	min	max	min
±0.5	31.6	33.5	30.1	32.2	37.1	35.4	36.7	34.2
±1.0	59.8	61.1	57.2	60.3	64.9	62.6	64.1	62.2
±1.5	75.8	76.6	75.2	76.4	80.1	77.8	79.5	77.5
±2.0	87.6	88.3	86.0	87.0	89.2	87.1	88.7	87.2
±2.5	91.3	92.7	91.8	92.2	93.8	91.7	93.6	91.8
±3.0	93.6	94.8	93.4	93.8	95.7	95.8	95.2	94.7
Max. Dev.	5.76	5.41	5.90	6.02	4.92	5.10	5.02	4.89
Avg. Dev.	1.08	1.06	1.10	1.07	0.95	1.02	0.97	1.03

VI. 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 temperature variations. This may help us to get a better insight into the temperature variation process. We try to do these now.

A. Online Feature Selection Net

There have been several [10], [11] attempts to use neural networks for feature selection. These methods are offline in nature. Here, we use an online feature selection method due to Pal and Chintalapudi [12]. We use the acronym FSMLP (e.g., for feature selection MLP) for the Pal–Chintalapudi modification of the standard MLP that can select features.

In a standard MLP, the effect of some features (inputs) can be eliminated by not allowing them into the network. If we can identify "partially useful" features, then they can be attenuated according to their relative usefulness. 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 opened. 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 gate function value, and the modulated feature value is then 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 learned using the training data.

Let $F : R \to [0, 1]$ be an attenuation (gate) function associated with an input node. 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 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 once γ_i is known, $F(\gamma_i)$ acts as a *fixed* multiplier for *all* input values of the ith feature.

The function F can have various forms. In the experiments described below, we use the attenuation function $F(\gamma) = 1/(1 + e^{-\gamma})$. Thus, the ith input node attenuates x_i by an amount $F(\gamma_i) \in (0, 1)$, where γ_i is a parameter to be learned during training. If $F(\gamma_i)$ is close to zero, we may choose to eliminate input feature x_i: this is how the FSMLP accomplishes feature selection. How do we learn γ_i? The backpropagation formulas for the MLP can simply be extended backward into this modified input layer to adjust the γ_is during training.

Let n_h = 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; $F : R \to (0, 1)$ = attenuator function with argument γ_i for input node i; $F'(\gamma_i)$ = derivative of F at γ_i; $w_{ji}^{ih}(t)$ = 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.

It can be easily shown [12] that the learning rule for connection weights remains the same for all layers except for $w_{ji}^{ih}(t)$. The update rules for $w_{ji}^{ih}(t)$ and γ_i are

$$w_{ji}^{ih}(t) = w_{ji}^{ih}(t-1) - \eta x_i \delta_j^1 F'(\gamma_i(t-1)) \tag{3}$$

$$\gamma_i(t) = \gamma_i(t-1) + \mu x_i \left(\sum_{j=1}^{n_h} w_{ji}^{ho} \delta_j^1 \right) F'(\gamma_i(t-1)). \tag{4}$$

The γ_i, $i = 1, 2, \ldots, p$ are initialized with values that make $F(\gamma_i)$ close to zero for all i. Consequently, $x_i F(\gamma_i)$ is small at

TABLE V
FSMLP ATTENUATION FACTORS FOR NETWORK WITH 15 INPUTS

No.	Attenuation	Feature	Remark
1.	99.37%	Max. pressure	Reject
2.	99.28%	Min. pressure	Reject
3.	62.06%	Max. vapor pressure	Accept
4.	70.18%	Min. vapor pressure	Accept
5.	73.80%	Max. relative humidity	Accept
6.	98.79%	Min. relative humidity	Reject
7.	50.67%	$T^{max}(t)$	Accept
8.	22.19%	$T^{min}(t)$	Accept
9.	90.53%	$T^{max}(t-2) - T^{max}(t-3)$	Reject
10.	99.54%	$T^{min}(t-2) - T^{min}(t-3)$	Reject
11.	99.71%	$T^{max}(t-1) - T^{max}(t-2)$	Reject
12.	90.62%	$T^{min}(t-1) - T^{min}(t-2)$	Reject
13.	81.13%	$T^{max}(t) - T^{max}(t-1)$	Accept
14.	70.02%	$T^{min}(t) - T^{min}(t-1)$	Accept
15.	62.24%	Rainfall	Accept

the beginning of training. So, 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 mean squared error is low or when the number of iterations reaches a maximum limit. Features with *low* attenuator weights are then eliminated from the feature set. In this investigation, we only consider those features whose attenuation values at the end of the training are less than 90%.

B. Results

In order to select the good features, we train the FSMLP using the entire dataset. And after the features are selected, we train the SOFM–MLP with the selected set of features. Table V displays the attenuation factors of the 15 features after training.

Table V reveals that only eight of the 15 features are important for prediction of the next day's temperature [$T^{max}(t)$ or $T^{min}(t)$]. The network rejects the maximum pressure, minimum pressure, and minimum relative humidity, but not the maximum relative humidity. It is very reasonable to expect that the maximum relative humidity can have influence on the temperature variation but not the minimum relative humidity. The network can capture these information. Similarly, of the six temperature gradients, the network picks up only the two most recent

TABLE VI
CUMULATIVE PERCENTAGE FREQUENCY TABLE FOR MLP AND SOFM–MLP USING SELECTED FEATURES

Range in °C	% Frequency of Temperature for test data							
	MLP				SOFM-MLP			
	$n_h = 10$		$n_h = 12$		$n_h = 10$		$n_h = 12$	
	max	min	max	min	max	min	max	min
± 0.5	54.5	34.0	51.0	33.5	39.0	36.5	37.5	35.0
± 1.0	61.0	62.5	60.0	62.0	65.5	62.5	64.5	62.0
± 1.5	77.0	78.0	76.0	76.5	82.0	77.5	80.5	78.5
± 2.0	87.5	88.0	87.0	87.0	89.5	88.0	89.0	87.5
± 2.5	92.0	91.5	91.0	91.5	94.0	93.0	93.5	92.5
± 3.0	94.5	95.5	93.5	94.0	96.5	95.0	96.0	95.0
Max. Dev.	5.59	4.92	4.86	5.06	4.35	4.35	4.63	4.02
Avg Dev.	1.06	1.03	1.07	1.06	0.91	1.00	0.91	1.01

temperature gradients, i.e., the difference of the maximum temperatures of today and yesterday [$T^{max}(t) - T^{max}(t-1)$] and the difference of the minimum temperatures of today and yesterday [$T^{min}(t) - T^{min}(t-1)$]. This tells us that only very local (with respect to time) variation of temperature has effect on predicting the temperature. Of the accepted features, FSMLP has given the maximum importance to the minimum temperature of today [$T^{min}(t)$]; the next important feature is the maximum temperature of today $T^{max}(t)$. This is also very logical, as we are predicting the maximum and minimum temperatures. The third most important feature selected by the network, as a meteorologist will expect, is the maximum vapor pressure of today.

In our earlier experiments, we did not use any validation set to guard against overtraining or memorization by the network. In order to demonstrate the fact that our earlier results do not suffer from overtraining and memorization, and to see the effect of validation, we now use a validation set. We now use 70 points for validation and 200 points for testing.

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

Table VI depicts the average performance of MLP and SOFM–MLP using the selected features in conjunction with a validation set. Since in these cases we have used only eight input features, we have restricted the maximum number of nodes in the hidden layer to 12 only.

Comparing Table VI with Table IV, we find that in this case, too, there is a marginal improvement in performance for SOFM–MLP with the selected features. Comparing the columns for SOFM–MLP with those of MLP in Table VI, we again find that SOFM–MLP outperforms the conventional MLP. The most important point is that we can use only a few features to get good results.

Fig. 3 depicts the plot of predicted maximum temperature averaged over ten runs corresponding to Table VI for the first 50

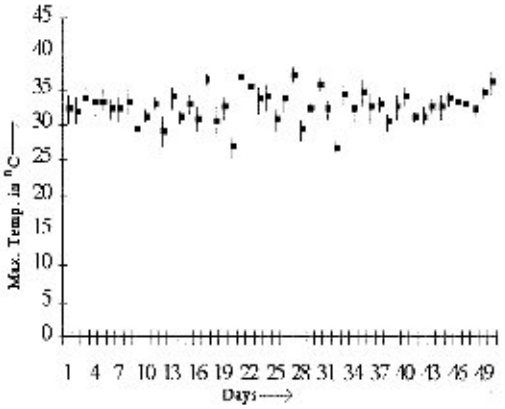

Fig. 3. Predicted maximum temperature (averaged over ten runs using SOFM–MLP) along with 6σ (-3σ, $+3\sigma$) length where σ is the standard deviation.

test points when $n_h = 10$. The vertical line indicates the length 6σ, (-3σ to $+3\sigma$) where σ is the standard deviation of the predictions for that particular day. In Fig. 3, we show only 50 points for visual clarity; the characteristics of the figure remain the same with all 200 points. Fig. 3 shows that the prediction by the SOFM–MLP network is quite consistent and is not influenced much by the initialization of the network.

VII. Conclusion and Discussion

We proposed a hybrid neural network model that combines the self-organizing feature map and the multilayer perceptron network for temperature prediction. In this context, we have used some derived features to enhance the prediction accuracy. The importance of feature analysis is further demonstrated using an online feature selection technique. The proposed method has been compared with both local and global predictors and has been found to produce a much better prediction than others.

To summarize our contribution and observations we can state the following.

1) Derived features could be more effective than raw features. For example, 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 over the raw features.
2) Feature selection is an important factor for better prediction of atmospheric parameters. In this regard, our FSMLP turns out to be an excellent tool that can select good features while learning the prediction task.
3) Nonlinear models such as MLP and RBF are better than the AR model.
4) Global predictor like MLP is found to perform better than the local predictors such as RBF network.
5) The proposed hybrid SOFM–MLP network consistently performs better than the conventional MLP network. Our comparison with another local predictor, the RBF network, further demonstrated the superiority of the proposed hybrid network.
6) The combined use of FSMLP and SOFM–MLP results in an excellent paradigm for prediction of atmospheric parameters.
7) There are couple of other areas where we need to do experiments. For example, we plan to use FSMLP and SOFM–MLP for prediction of other atmospheric parameters.

We would also like to investigate their usefulness in different pattern recognition problems.

Acknowledgment

The authors are grateful to S. K. Banerjee (IMD, Calcutta) for providing the data and suggestions.

References

[1] D. Tsintikidis, J. L. Haferman, E. N. Anagnostou, W. F. Krajewski, and T. F. Smith, "A neural network approach to estimating rainfall from spaceborne microwave data," *IEEE Trans. Geosci. Remote Sensing*, vol. 35, pp. 1079–1093, Sept. 1997.
[2] H. Salehfar and S. A. Benson, "Electric utility coal quality analysis using artificial neural network techniques," *Neurocomputing*, vol. 23, pp. 195–206, 1998.
[3] S. Haykin, *Neural Networks a Comprehensive Foundation*, 2nd ed, Singapore: Pearson Education, 1999.
[4] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automatic Control*, vol. AC-19, pp. 716–723, 1974.
[5] T. Kohonen, K. Torkkola, M. Shozakai, J. Kangas, and O. Venta, "Microprocessor implementation of a large vocabulary speech recognizer and phonetic typewriter for Finnish and Japanese," in *Proc. Eur. Conf. Speech Technology*, Edinburg, U.K., 1987, pp. 377–380.
[6] N. M. Nasarabadi and Y. Feng, "Vector quantization of images based on kohonen self-organizing feature maps," *Proc. IEEE Int. Conf. on Neural Networks*, pp. I-101–I-108, 1988.
[7] K. M. Marks and K. F. Goser, "Analysis of VLSI process data based on self-organizing feature maps," in *Proc. Neuro-Nimes*, Nimes, France, pp. 337–347.
[8] Z. Chi, J. Wu, and H. Yan, "Handwritten numeral character recognition using self-organizing maps and fuzzy rules," *Pattern Recognit.*, vol. 28, no. 1, pp. 59–66, 1995.
[9] T. Kohonen, *The Self-Organizing Maps*, 2nd ed. Berlin, Germany: Springer-Verlag, 1997.
[10] D. W. Ruck, S. K. Rogers, and M. Kbrisky, "Feature selection using a multilayer perceptron," *J. Neural Network Comput.*, vol. 2, no. 2, pp. 40–48, 1990.
[11] R. De, N. R. Pal, and S. K. Pal, "Feature analysis: Neural network and fuzzy set theoretic approach," *Pattern Recognit.*, vol. 30, no. 10, pp. 1579–1590, 1997.
[12] N. R. Pal and K. Chintalapudi, "A connectionist system for feature selection," *Neural, Parallel Sci. Computat.*, vol. 5, no. 3, pp. 359–381, 1997.

Nikhil R. Pal (M'91–SM'00) received the B.Sc. and M.B.M. degrees from the University of Calcutta, Calcutta, India, and the M.Tech. and Ph.D. degrees from the Indian Statistical Institute (ISI), Calcutta.

He is currently a Professor with the ISI. He is an Associate Editor of the *International Journal of Fuzzy Systems* and the *International Journal of Approximate Reasoning*. He is an Area Editor of *Fuzzy Sets and Systems*. He is on the editorial advisory board of the *International Journal of Knowledge-Based Intelligent Engineering Systems* and is a Steering Committee member of *Applied Soft Computing*.

Dr. Pal is an Associate Editor for the IEEE Transactions on Fuzzy Systems and the IEEE Transactions on Systems, Man, and Cybernetics B. He is an Independent Theme Chair of the World Federation of Soft Computing and a governing board member of the Asia Pacific Neural Net Assembly. He was the Program Chair of the *4th International Conference on Advances in Pattern Recognition and Digital Techniques* (December 1999, Calcutta, India) and the General Chair of *2002 AFSS International Conference on Fuzzy Systems* (Calcutta, 2002).

Srimanta Pal received the B.Sc. degree from the University of Calcutta, India, the B.Tech. and M.B.A. degrees from the Jadavpur University, Calcutta, the M.Tech. degree from the Indian Statistical Institute (ISI), Calcutta, and the Ph.D. degree from the Indian Institute Technology, Kharagpur.

He is currently an Associate Professor with ISI. His research interests include computational intelligence and nonstandard computing. He has worked as a Co-Guest Editor for a 2002 special issue on "Computational Intelligence for Pattern Recognition" in the *International Journal of Pattern Recognition and Artificial Intelligence*.

Jyotirmoy Das received the M.Tech. and Ph.D. degrees from the University of Calcutta, Calcutta, India.

He is currently the Senior Professor and the Head of the Electronics and Communication Sciencies Unit, Indian Statistical Institute, Calcutta. His current research interests include atmospheric science problems, computational intelligence, and millimeterwave propagation. He has coauthored *Digital Memory Technology* (New Delhi, India: Wiley Eastern, Ltd., 1990).

Dr. Das is a Fellow of the National Academy of Science.

Kausik Majumdar is currently pursuing the B.E. degree at Jadavpur University, Calcutta, India.