INFORMATION SCIENCES **76**, 297–337 (1994)

Fuzzy Versions of Kohonen's Net and MLP-Based Classification: Performance Evaluation for Certain Nonconvex Decision Regions

SANKAR K. PAL*

and

SUSHMITA MITRA

Machine Intelligence Unit, Indian Statistical Institute, 203, B. T. Road, Calcutta 700035, India

Communicated by Abe Kandel

ABSTRACT

Classification of certain linearly nonseparable pattern classes with nonconvex decision regions is a problem that cannot be efficiently handled by the Bayes' classifier for normal distributions or other metric-based methods. An attempt is made here to demonstrate the ability of fuzzy versions of Kohonen's net and the multilayer perceptron for classification of such patterns. In these models, the uncertainties involved in the input description and output decision have been taken care of by the concept of fuzzy sets whereas the neural net theory helps to generate the required concave and/or disconnected decision regions. Superiority of these fuzzy models (over the respective conventional versions, the Bayes' classifier and seven other existing neural algorithms) has been adequately established when they are implemented on different sets of linearly nonseparable pattern classes. The effect of fuzzification at the input has been investigated for both models. The contribution of the a priori probabilities of the pattern classes in the back-propagation procedure for weight updating has also been studied.

1. INTRODUCTION

Artificial neural networks [1]–[3] are massively parallel interconnections of simple neurons that function as a collective system. An advantage of neural nets lies in their high computation rates provided by massive parallelism, so that real-time processing of huge data sets becomes feasible with proper hardware. Information is encoded among the various connection weights in a distributed manner.

The utility of fuzzy sets [4]–[8] lies in their capability, to a reasonable extent, to model uncertain or ambiguous data so often encountered in real life. Besides, human reasoning is somewhat fuzzy in nature. Hence, to enable a system to tackle ambiguous (ill-defined) data in an effective manner, one may incorporate the concept of fuzzy sets into the neural network.

There have been several attempts recently [9]–[12] to make a fusion of fuzzy logic and neural networks for better performance in decisionmaking systems. The concept of fuzzy sets takes care of the uncertainties involved in the input description and also in the output decision whereas the neural net model helps to generate the appropriate class boundaries (which may be disconnected and concave). The development of fuzzy versions of Kohonen's self-organizing network [13] and the multilayer perceptron (MLP) [14] for pattern recognition recently has been reported. Their suitability for classification of speech patterns has also been demonstrated.

During self-organization, the input vector of the fuzzy extension to Kohonen's model [13] includes some contextual information (with lower weightage) regarding the class membership of the pattern (in addition to the input feature information). This may, therefore, be termed a *partially supervised* fuzzy classifier. The training samples are presented to the network in cycles until the output space is *ordered*, as measured by an *index of disorder*. Next the neurons in the output space are *calibrated* (labeled) by using only the class information parts of these input vectors. The self-organization and calibration together constitute the training phase for the fuzzy model.

The fuzzy multilayer perceptron (MLP) [14], on the other hand, functions as a *fully supervised* classifier. In the learning phase the training samples are presented to the network in cycles until it finally converges to a minimum error solution. A heuristic for gradually decreasing the learning rate and momentum is used to help prevent oscillations of the mean square error in the process of convergence.

Both proposed fuzzy neural models [13], [14] are capable of handling input features in quantitative and/or linguistic form and can take care of uncertainty and/or impreciseness, to a reasonable extent, in the input specifications as well as in the output decision. The components of the input vector consist of the membership values to the overlapping partitions of linguistic properties *low*, *medium*, and *high* corresponding to each input feature. Thereby an n-dimensional feature space is decomposed into 3^n overlapping subregions corresponding to the three primary properties [15]. This enables the models to utilize more local information of the feature space and is found to be more suitable in handling linearly nonseparable pattern classes having nonconvex decision regions. Output is provided in terms of membership functions. One also can obtain hard decisions (as a special case).

In this work we concentrate on the problem of handling pattern classes that have concave and disconnected decision regions. Such patterns cannot be properly classified by the parametric Bayesian method for normal distributions or other metric-based algorithms. Three sets of such linearly nonseparable pattern classes (artificially generated) are depicted in Figures 1–3. There are two pattern classes (1 and 2) in each case. The region of *no pattern points* is modeled as the class *none* (*no class*). In two cases, the decision region of class 2 is disjoint and its a priori probability is much lower as compared to that of the other classes. It is to be noted that the Gaussian classifier makes strong assumptions concerning underlying distributions and is more appropriate when the class distributions are known and match the Gaussian assumptions [1]. Nonparametric classifiers generally construct a measure of performance over the training set and adjust the variables of the classifier to optimize this measure [16]. Neural networks belong to this category of classifiers and possess the ability to also perform nonlinear classification involving nonconvex and disjoint decision

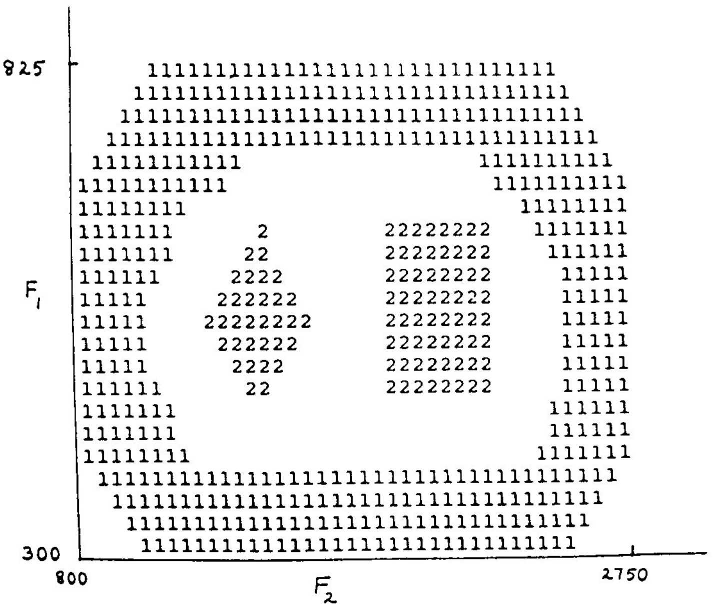

Fig. 1. *Pattern Set A* in the F_1-F_2 plane.

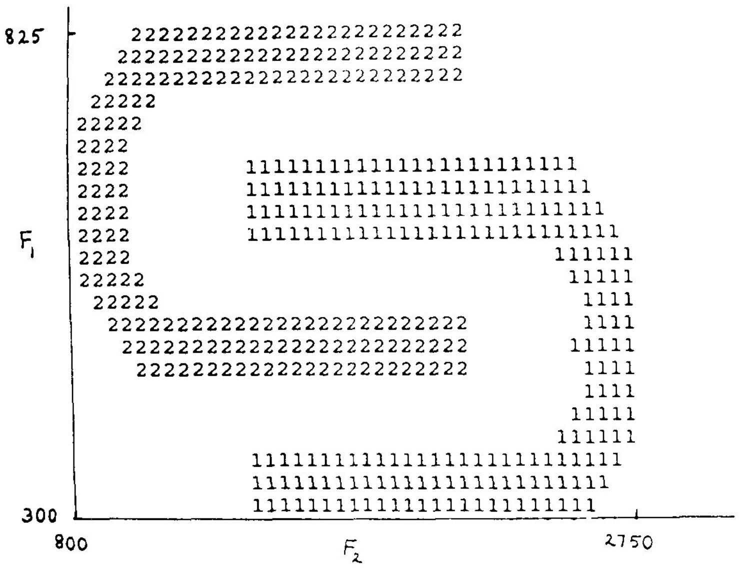

Fig. 2. *Pattern Set B* in the F_1-F_2 plane.

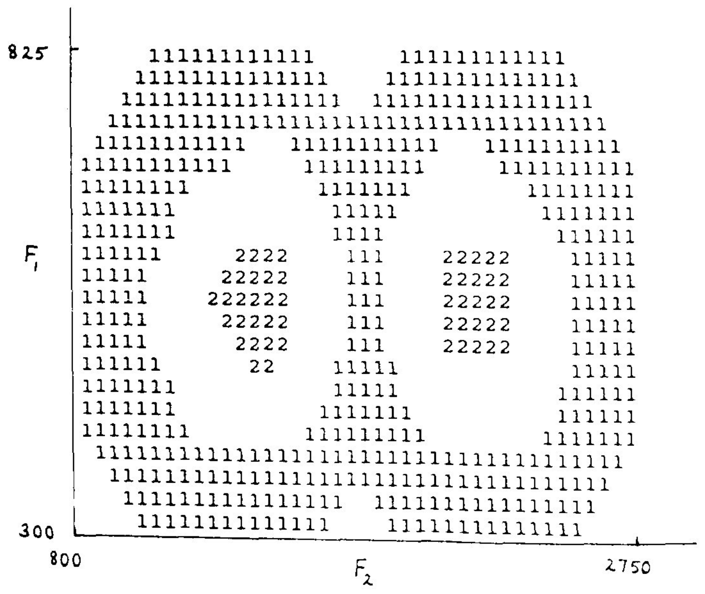

Fig. 3. *Pattern Set C* in the F_1-F_2 plane.

regions. It may be mentioned in this connection that the nonparametric single layer perceptrons are incapable of classifying linearly nonseparable patterns [17].

In this paper, an attempt is made to demonstrate the ability of the fuzzy versions of Kohonen's self-organizing network [13] and the multilayer perceptron [14] for classification of the aforesaid linearly nonseparable patterns. The effect of fuzzification at the input has been investigated for both models. Their performance is adequately compared with those of the Bayes' classifier, the nonfuzzy versions of the two neural models, other existing neural algorithms of Rumelhart and McClelland [2], McClelland (reported in [18]), Franzini [18], Chan and Fallside [19], Tollenaere [20], Silverman [21], and networks using second-order weight correction [2]. The contribution of the a priori probabilities of the pattern classes in the error derivative of the back-propagation procedure for weight updating has also been studied.

2. KOHONEN'S SELF-ORGANIZING NEURAL MODEL

Let us consider the self-organizing Kohonen network [3]. Let M input signals be simultaneously incident on each of an $N \times N$ array of neurons. The output of the ith neuron is defined as

$$\eta_i(t) = \sigma\left[[\mathbf{m}_i(t)]^T \mathbf{x}(t) + \sum_{k \in S_i} w_{ki} \eta_k(t - \Delta t) \right], \tag{1}$$

where $\mathbf{x}$ is the M-dimensional input vector incident on the neuron along the connection weight vector $\mathbf{m}_i$, k belongs to the subset S_i of neurons having interconnections with the ith neuron, w_{ki} denotes the fixed feedback coupling between the kth and ith neurons, $\sigma[\cdot]$ is a suitable sigmoidal output function, t denotes a discrete time index, and T stands for the transpose.

If the best match between vectors $\mathbf{m}_i$ and $\mathbf{x}$ occurs at neuron c, then we have

$$\|\mathbf{x} - \mathbf{m}_c\| = \min_i \|\mathbf{x} - \mathbf{m}_i\|, \qquad i = 0, 1, \ldots, N^2, \tag{2}$$

where $\|\cdot\|$ indicates the Euclidean norm.

The weight updating is given by [3] as

$$\mathbf{m}_i(t+1)=\begin{cases}\mathbf{m}_i(t)+\alpha(t)(\mathbf{x}(t)-\mathbf{m}_i(t)), & \text{for } i\epsilon N_c,\\ \mathbf{m}_i(t), & \text{otherwise,}\end{cases} \tag{3}$$

where $\alpha(t)$ is a positive constant that decays with time and N_c defines a topological neighborhood around the maximally responding neuron c, such that it also decreases with time. After a number of sweeps through the training data during self-organization, with weight updating at each iteration obeying Eq. (3), the asymptotic values of $\mathbf{m}_i$ cause the output space to attain proper topological ordering.

3. KOHONEN'S NET AS A FUZZY CLASSIFIER

In this section we describe, in brief, a fuzzy version of Kohonen's model (reported in [13]) that is capable of representing input in linguistic and/or quantitative form and providing output decision in terms of membership values. We consider a single-layer two-dimensional rectangular array of neurons with short-range lateral feedback interconnections between neighboring units. In the first stage a set of training data is used by the network to initially self-organize the connection weights and finally *calibrate* the output space. After a number of sweeps through the training set, the output space becomes appropriately ordered. An *index of disorder* is computed to evaluate a measure of this ordering. The network is now supposed to encode the input space information among its connection weights.

A. THE INPUT VECTOR

The input to the proposed neural network model consists of two portions. In addition to the input feature representation in linguistic form, there is some contextual information regarding the fuzzy class membership of each pattern (used as training data) during self-organization of the network.

1. Feature Information

Let $X=\{X_1,X_2,\ldots,X_L\}$ be a set of L pattern points in an n-dimensional feature space. In the model under consideration, each input feature

F_j (in quantitative and/or linguistic form) can be expressed in terms of membership values indicating a measure of belongingness to each of the linguistic properties *low*, *medium*, and *high* [15]. Therefore, an n-dimensional pattern $\mathbf{X}_i = \mathbf{F}_i = [F_{i1}, F_{i2}, \ldots, F_{in}]$ may be represented as a $3n$-dimensional vector

$$\mathbf{X}_i = \left[\mu_{low(F_{i1})}(\mathbf{X}_i), \mu_{medium(F_{i1})}(\mathbf{X}_i), \mu_{high(F_{i1})}(\mathbf{X}_i), \ldots, \mu_{high(F_{in})}(\mathbf{X}_i) \right], \quad (4)$$

where the μ value indicates the membership of $\mathbf{X}$ to the corresponding linguistic set along each feature axis.

It is to be noted here than an n-dimensional feature space is decomposed into 3^n overlapping subregions corresponding to the three primary properties [15]. This enables the model to utilize more local information of the feature space and it seems to be more effective in handling the linearly nonseparable pattern classes (shown in Figures 1–3) that have nonconvex decision regions.

For numeric feature value F_j (along the jth axis), the π function (in the one-dimensional form) lying in the range $[0,1]$ is given as

$$\pi(F_j; c, \lambda) = \begin{cases} 2(1 - |F_j - c|/\lambda)^2, & \text{for } \lambda/2 \leqslant |F_j - c| \leqslant \lambda, \\ 1 - 2(|F_j - c|/\lambda)^2, & \text{for } 0 \leqslant |F_j - c| \leqslant \lambda/2, \\ 0, & \text{otherwise}, \end{cases} \quad (5)$$

where $\lambda > 0$ is the radius of the π function with c as the central point at which $\pi(c; c, \lambda) = 1$. The values of the λs and cs for each of the three linguistic properties *low*, *medium*, and *high* are given by Eqs. (7)–(9).

When the input feature F_j is linguistic, its membership values for the π sets *low*, *medium*, and *high* in Eq. (4) may be quantified as

$$\begin{aligned} low &\equiv \left\{ \frac{0.95}{L}, \frac{0.6}{M}, \frac{0.02}{H} \right\}, \\ medium &\equiv \left\{ \frac{0.7}{L}, \frac{0.95}{M}, \frac{0.7}{H} \right\}, \\ high &\equiv \left\{ \frac{0.02}{L}, \frac{0.6}{M}, \frac{0.95}{H} \right\}. \end{aligned} \quad (6)$$

Let $F_{j_{\max}}$ and $F_{j_{\min}}$ denote the upper and lower bounds of the dynamic range of feature F_j in all L pattern points, considering numerical values only. Then for the three linguistic property sets we define

$$\lambda_{medium(F_j)} = \frac{1}{2}\left(F_{j_{\max}} - F_{j_{\min}}\right)$$

$$c_{medium(F_j)} = F_{j_{\min}} + \lambda_{medium(F_j)} \tag{7}$$

$$\lambda_{low(F_j)} = \frac{1}{fdenom}\left(c_{medium(F_j)} - F_{j_{\min}}\right)$$

$$c_{low(F_j)} = c_{medium(F_j)} - 0.5 * \lambda_{low(F_j)} \tag{8}$$

$$\lambda_{high(F_j)} = \frac{1}{fdenom}\left(F_{j_{\max}} - c_{medium(F_j)}\right)$$

$$c_{high(F_j)} = c_{medium(F_j)} + 0.5 * \lambda_{high(F_j)}, \tag{9}$$

where $0.5 \leqslant fdenom \leqslant 1.0$ is a parameter that controls the extent of overlapping.

2. *Class Information in Contextual Form*

In many real-life problems, the data are generally ill-defined with overlapping or fuzzy class boundaries. Each pattern used in training may possess finite belongingness to more than one class. To model such data, it often becomes necessary to incorporate some contextual information regarding class membership as part of the input vector [13]. However, during self-organization this part of the input vector is assigned a lower weightage so that the linguistic feature properties dominate in determining the ordering of the output space. During calibration we use the contextual class membership information part of the input vector only for determining the *hard* labeling of the output space.

The pattern $\mathbf{X}_i$ is considered to be a concatenation of the linguistic properties in Eq. (4) and the contextual information regarding class membership. Let the input vector be expressed as

$$\mathbf{x} = [\mathbf{x}', \mathbf{x}'']^T = [\mathbf{x}', 0]^T + [0, \mathbf{x}'']^T, \tag{10}$$

where $\mathbf{x}'$ contains the linguistic information in the $3n$-dimensional space of Eq. (4) and $\mathbf{x}''$ covers the class membership information in an l-dimensional space for an l-class problem domain. So the input vector $\mathbf{x}$ lies in a

$(3n+l)$-dimensional space. Both $\mathbf{x}'$ and $\mathbf{x}''$ are expressed as membership values.

The membership of the ith pattern to class C_k is defined as

$$\mu_k(\mathbf{X}_i) = \frac{1}{1+(z_{ik}/F_d)^{F_e}}, \tag{11}$$

where z_{ik} is the weighted distance between the ith pattern and the kth class, $0 \leqslant \mu_k(\mathbf{X}_i) \leqslant 1$ and the positive constants F_d and F_e are the denominational and exponential fuzzy generators that control the amount of fuzziness in this class-membership set.

In this connection, note that the particular problem of modeling the decision surfaces for classifying the linearly nonseparable patterns of Figures 1–3 might also be eased by incorporating the contextual class information in the input vector. However, in such cases, z_{ik} of Eq. (11) may be set to 0 for a particular class and to *infinity* for the remaining classes, so that $\mu_k(\mathbf{X}_i) = \{0,1\}$.

For the ith input pattern we define

$$\mathbf{x}'' = s * [\mu_1(\mathbf{X}_i), \ldots, \mu_l(\mathbf{X}_i)]^T, \tag{12}$$

where $0 < s \leqslant 1$ is the scaling factor. To ensure that the norm of the linguistic part $\mathbf{x}'$ predominates over that of the class membership part $\mathbf{x}''$ in Eq. (10) during self-organization, we choose $s < 0.5$.

3. *Modification of Input during Calibration*

During calibration of the output space the input vector chosen is $\mathbf{x} = [0, \mathbf{x}'']$, where $\mathbf{x}''$ is given by Eq. (12), such that

$$\mu_q(\mathbf{X}_i) = \begin{cases} 1, & \text{if } q = k, \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

for $k \in \{1, \ldots, l\}$ and $s = 1$. The N^2 neuron outputs η_i are calibrated w.r.t. the l classes. Here the class information of the training patterns is given full weightage while the input feature information is suppressed. The primary objective of this stage is to label each neuron by the partition for which it elicits the maximum response. The resulting *hard* (labeled) partitioning of the output space may be used to qualitatively assess the topological ordering of the pattern classes w.r.t. the input feature space. We also generate a *fuzzy* partitioning of the output space.

B. THE ALGORITHM

Consider an $N \times N$ array of neurons such that the output of the ith neuron is given by Eq. (1), with the subset S_i of neurons being defined as its r-neighborhood N_r, where $0 \leqslant r \leqslant 3$. We use

$$w_{ki} = \begin{cases} b, & \text{for } r = 1, \\ -\dfrac{b}{2}, & \text{for } r = 2, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Here b is the mutual interaction weight for the lateral coupling w_{ki}.

1. *Weight Updating*

Initially the components of the $\mathbf{m}_i$s are set to small random values lying in the range [0,0.5]. Let the best match between vectors $\mathbf{m}_i$ and $\mathbf{x}$, selected using Eq. (2), occur at neuron c. Using Eq. (3), the weight updating expression may be stated as

$$\mathbf{m}_i(t+1) = \begin{cases} \mathbf{m}_i(t) + h_{ci} * (\mathbf{x}(t) - \mathbf{m}_i(t)), & \text{for } i \in N_r,\ r = 0, 1, \ldots, 3, \\ \mathbf{m}_i(t), & \text{otherwise,} \end{cases} \tag{15}$$

where N_r describes a r-neighborhood around neuron c such that r decreases with time. Here the gain factor h_{ci} is considered to be *bell-shaped* like the π function, such that $|h_{ci}|$ is the largest when $i = c$ and gradually decreases to zero with increasing distance from c. Besides, $|h_{ci}|$ also decays with time.

We define

$$h_{ci} = \frac{(1 - r * f) * \alpha'}{\left[1 + (nt/cdenom)^2\right]}, \tag{16}$$

where nt is the number of sweeps already made through the entire set of training samples at any point of time, *cdenom* is a positive constant suitably chosen and $0 < f$, $\alpha' < 1$. The decay of $|h_{ci}|$ with time is controlled by nt. The slowly decreasing radius of the *bell-shaped* function h_{ci} and the corresponding change in $|h_{ci}|$ are controlled by the parameters r and f. Due to the process of self-organization, the randomly chosen initial $\mathbf{m}_i$s gradually attain new values according to Eqs. (2) and (15) such that the output space acquires appropriate topological ordering.

2. *Index of Disorder*

An index of disorder D may be defined to provide a measure of this ordering. Let *msd* denote the mean square distance between the input vector and the weight vectors in the r-neighborhood of neuron c. We define

$$msd = \frac{1}{|trainset|} \sum_{\mathbf{x} \in trainset} \left[\sum_{r=0}^{3} \left\{ \left(\frac{1}{|N_r|} \sum_{i \in N_r} \|\mathbf{x} - \mathbf{m}_i\|^2 \right) * (1 - r * f) \right\} \right], \quad (17)$$

where $|trainset|$ refers to the number of input pattern vectors in the training set. This definition ensures that neurons nearer c (smaller r) contribute more to *msd* than those farther away. Also

$$f = \begin{cases} \frac{1}{4}, & 0 \leqslant r \leqslant 3, \quad \text{for } ncnt = 1, \\ \frac{1}{3}, & 0 \leqslant r \leqslant 2, \quad \text{for } ncnt = 2, \\ \frac{1}{2}, & 0 \leqslant r \leqslant 1, \quad \text{otherwise}. \end{cases} \quad (18)$$

Here $|N_r|$ denotes the number of neurons in the r-neighborhood of neuron c such that $|N_1| \leqslant 8$, $|N_2| \leqslant 16$, and $|N_3| \leqslant 24$ depending upon the position of c in the two-dimensional array. Note that N_0 implies the neuron c itself.

The expression for the index of disorder is given as

$$D = msd(nt - kn) - msd(nt), \quad (19)$$

where $msd(nt)$ denotes the mean square distance by Eq. (17) at the end of the ntth sweep through the training set and kn is a suitable positive integer. Further, D is sampled at intervals of kn sweeps. Initially *ncnt* is set to 1. Then

$$ncnt = \begin{cases} ncnt + 1, & \text{if } D < \delta, \\ ncnt, & \text{otherwise,} \end{cases} \tag{20}$$

where $0 < \delta \leqslant 0.001$. The process is terminated when $ncnt > 3$, so that always $r \geqslant 1$ in Eq. (18). For good self-organization, the value of *msd* and, therefore, D should gradually decrease. It is to be noted that r and f of Eqs. (15) and (16), that control $|h_{ci}|$, obey Eq. (18). The parameter *ncnt* of Eq. (20) depending on D of Eq. (19), controls r and f of Eq. (18).

3. *Partitioning during Calibration*

During calibration the input vector $\mathbf{x} = [0, \mathbf{x}'']$ of Eq. (10) is applied to the neural network. Let the $(i1)_k$th neuron generate the highest output η_{f_k} for class C_k. We define a membership value for the output of neuron i when calibrated for class C_k simply as

$$\mu_k(\eta_i) = \frac{\eta_{i_k}}{\eta_{f_k}}, \quad \text{for } i = 1, \ldots, N^2 \text{ and } k = 1, \ldots, l, \tag{21}$$

such that $0 \leqslant \mu_k(\eta_i) \leqslant 1$ and $\mu_k(\eta_i) = 1$ for $i = (i1)_k$.

Each neuron i may be marked by the output class C_k, among all l classes, that elicits the maximal response η_{i_k}. This generates a hard partitioning of the output space. On the other hand, each neuron i has a finite belonging or output membership $\mu_k(\eta_i)$ to class C_k by Eq. (21). We may generate *crisp* boundaries for the fuzzy partitioning of the output space by considering for each of the l classes the *alpha-cut* set $\{i | \mu_k(\eta_i) > \alpha'\}$, $0 < \alpha' \leqslant 1$, where α' is a suitably chosen value. An ordered map of the output space indicates good self-organization and hence grouping of the patterns according to similarity.

4. *Testing Phase*

After self-organization, the proposed model encodes all input data information, along with the corresponding contextual class membership values, distributed among its connection weights. During calibration, the neurons are labeled by the pattern classes and the corresponding membership values also are assigned. This is the desired fuzzy classifier. In the final stage, a separate set of test patterns is supplied as input to the neural network model and its performance is evaluated.

During this phase the input test vector $\mathbf{x} = [\mathbf{x}', 0]^T$, consisting of only the linguistic information in the $3n$-dimensional space defined by Eq. (4), is applied to the network. Let the $p1$th and $p2$th neurons generate the highest and second highest outputs η_{f_p} and η_{s_p}, respectively, for test pattern $\mathbf{p}$. We define

$$\begin{aligned} \mu_{k_1}(\eta_{f_{pm}}) &= \mu_{k1}(\eta_{p1}), \\ \mu_{k_2}(\eta_{s_{pm}}) &= \frac{1}{\eta_{f_p}} \mu_{k2}(\eta_{p2}) * \eta_{s_p}, \end{aligned} \tag{22}$$

and $k_1 = k1$, $k_2 = k2$, if $\mu_{k1}(\eta_{p1}) \geqslant (1/\eta_{f_p})\mu_{k2}(\eta_{p2}) * \eta_{s_p}$. Otherwise,

$$\begin{aligned} \mu_{k_1}(\eta_{f_{pm}}) &= \frac{1}{\eta_{f_p}} \mu_{k2}(\eta_{p2}) * \eta_{s_p}, \\ \mu_{k_2}(\eta_{s_{pm}}) &= \mu_{k1}(\eta_{p1}), \end{aligned} \tag{23}$$

such that $k_1 = k2$ and $k_2 = k1$. Here $k1$ and $k2$ refer to the output classes (hard partitions) C_{k1} and C_{k2} that elicited maximal strength responses at the $p1$th and $p2$th neurons, respectively, during calibration. On the other hand, C_{k_1} and C_{k_2} are dependent both on the actual output responses during testing and the membership values evaluated during calibration w.r.t. classes C_{k1} and C_{k2}. The membership values on the right-hand side of Eqs. (22) and (23) are defined as

$$\mu_{k1}(\eta_{p1}) = \frac{\eta_{(p1)_{k1}}}{\eta_{f_{k1}}} \tag{24}$$

from Eq. (21), where $\eta_{f_{k1}}$ and $\eta_{(p1)_{k1}}$ are obtained during calibration for class C_{k1}. Hence pattern $\mathbf{p}$ may be classified as belonging to class C_{k_1} with membership $\mu_{k_1}(\eta_{f_{pm}})$ lying in the interval $[0, 1]$, using the first choice and to class C_{k_2} with membership $\mu_{k_2}(\eta_{s_{pm}})$ using the second choice. It is to be noted that classes C_{k_1} and C_{k_2} are determined from classes C_{k1} and C_{k2} by Eqs. (22) and (23).

4. MULTILAYER PERCEPTRON

Let us next consider the MLP [2] network. After a lowermost input layer there are usually nay number of intermediate or *hidden* layers followed by an output layer at the top. There exist no interconnections within a layer whereas all neurons in a layer are fully connected to neurons in adjacent layers. The total input x_j^{h+1} received by neuron j in layer $h+1$ is defined as

$$x_j^{h+1} = \sum_i y_i^h w_{ji}^h - \theta_j^{h+1}, \tag{25}$$

where y_i^h is the state of the ith neuron in the preceding hth layer, w_{ji}^h is the weight of the connection from the ith neuron in layer h to the jth neuron in layer $h+1$ and θ_j^{h+1} is the threshold of the jth neuron in layer $h+1$. Threshold θ_j^{h+1} may be eliminated by giving the unit j in layer $h+1$ an extra input line with a fixed activity level of 1 and a weight of $-\theta_j^{h-1}$.

The output of a neuron in any layer other than the input layer ($h > 0$) is a monotonic nonlinear function of its total input and is given as

$$y_j^h = \frac{1}{1+e^{-x_j^h}}. \tag{26}$$

For nodes in the input layer, we have $y_j^0 = x_j^0$, where x_j^0 is the jth component of the input vector clamped at the input layer.

The least mean square (LMS) error in output vectors, for a given network weight vector $\mathbf{w}$, is defined as

$$E(\mathbf{w}) = \tfrac{1}{2} \sum_{j,c} \left(y_{j,c}^H(\mathbf{w}) - d_{j,c}\right)^2, \tag{27}$$

where $y_{j,c}^H(\mathbf{w})$ is the state obtained for output node j in layer H in input–output case c and $d_{j,c}$ is its desired state specified by the teacher. One method for minimization of $E(\mathbf{w})$ is to apply the method of gradient descent by starting with any set of weights and repeatedly updating each weight by an amount

$$\Delta w_{ji}^h(t) = -\epsilon \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}^h(t-1), \tag{28}$$

where the positive constant ϵ controls the descent, $0 \leqslant \alpha \leqslant 1$ is the damping coefficient or momentum, and t denotes the number of the iteration currently in progress.

Using Eqs. (25)–(27), we have

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial_{yj}} y_j^h \left(1 - y_j^h\right) y_i^{h-1}, \tag{29}$$

where

$$\frac{\partial E}{\partial y_j} = \begin{cases} y_j^H - d_j, & \text{for } h = H, \\ \sum\limits_k \dfrac{\partial E}{\partial y_k} \dfrac{dy_k}{dx_k} w_{kj}^h, & \text{otherwise}, \end{cases} \tag{30}$$

such that units j and k lie in layers h and $h+1$, respectively. The central idea is to first use a forward pass for each input–output case c, starting at the input neurons, to compute the activity levels of all the neurons in the network. Then a backward pass, starting at the output neurons, is used to compute the error derivative $\partial E / \partial y_j$ and back-propagate to enable weight updating until the input layer is reached.

After a number of sweeps through the training data, the error $E(\mathbf{w})$ in Eq. (27) may be minimized. In the testing phase the neural net is expected to be able to utilize the information encoded in its connection weights to assign the correct output labels for the test vectors that are now clamped only at the input layer.

5. FUZZY EXTENSION TO THE MLP MODEL

We now discuss, in brief, the fuzzy version of the MLP (reported in [14]), which is capable of representing input in linguistic and/or quantitative form and providing output decision in terms of membership values. The components of the input vector consist of the membership values to the overlapping partitions of linguistic properties *low*, *medium*, and *high* corresponding to each input feature. This provides scope for incorporating linguistic information in both the training and testing phases of the said model and increases its robustness in tackling imprecise or uncertain input specifications. During training, supervised learning is used to assign output membership values lying in the range $[0, 1]$ to the training vectors. The back-propagated error has inherently more weightage in case of nodes with higher membership values such that the contribution of ambiguous or uncertain vectors to the weight correction is automatically reduced. A heuristic for gradually decreasing the learning rate and the momentum is used to help avoid spurious local minima and usually prevent oscillations of the mean square error in the weight space, in the process of convergence to a minimum error solution.

The details regarding the input feature representation in this model (for quantitative and/or linguistic input) is the same as explained earlier in Section 3A with reference to the fuzzy self-organizing model. The input vector $\mathbf{x}$ is represented in the $3n$-dimensional space of Eq. (4).

A. OUTPUT VECTOR REPRESENTATION

For an l-class problem domain, the membership of the ith pattern to class C_k, lying in the range $[0,1]$ is given as in Eq. (11). Then for the ith input pattern we define the desired output of the jth output node as

$$d_j = \mu_j(\mathbf{X}_i), \tag{31}$$

where $d_j = [0,1]$. During testing, the output of the jth output neuron denotes the inferred membership value of a test pattern to the jth class.

Note that in the special case of classifying the linearly nonseparable patterns of Figures 1–3, the weighted distance z_{ik} of Eq. (11) between the ith pattern and the kth class is set to 0 for one class and to *infinity* for the remaining classes. Therefore, $d_j = \mu_j(\mathbf{X}_i) = \{0,1\}$ in such cases.

B. WEIGHT UPDATING

The ϵ of Eq. (28) is gradually decreased in discrete steps, taking values from the chosen set $\{2,1,0.5,0.3,0.1,0.05,0.01,0.005,0.001\}$, while the momentum factor α is also decreased. Let the various values of ϵ be indicated by $\epsilon_0 = 2$, $\epsilon_1 = 1, \ldots, \epsilon_q = 0.001$ such that ϵ_i indicates the $(i+1)$th value of ϵ. Let $\alpha_0 = 0.9$ and $\alpha_1 = \alpha_2 = \cdots = \alpha_q = 0.5$. Note that α close to zero is avoided because small values of α are unable to prevent unwanted oscillations. We use

$$i = \begin{cases} i+1, & \text{if } mse(nt-kn) - mse(nt) < \delta, \\ i, & \text{otherwise}, \end{cases} \tag{32}$$

where $i=0$ initially, $|\epsilon| = q+1$, and $0 < \delta \leqslant 0.0001$. Here $mse(nt)$ is the mean square error at the end of the ntth sweep through the training set and kn is a positive integer such that mse is sampled at intervals of kn sweeps. The process is terminated when $i > q$ and $\epsilon_q = 0.001$. At this stage the network is said to have converged to a *good* minimum error solution and the corresponding value of nt indicates the number of sweeps required in the process.

We use two measures of percent correct classification for the training set. The output, after a number of updating steps, is considered a *perfect match* if the value of each output neuron y_j^H is within a margin of 0.1 of the desired membership value d_j. This is a *stricter* criterion than the *best match*, where we test whether the jth neuron output y_j^H has the maximum activation when the jth component d_j of the desired output vector also has the highest value, provided $y_j^H > 0.5$.

6. OTHER NEURAL ALGORITHMS FOR COMPARISON

In this section we briefly describe the salient features of the other neural algorithms in the comparison. These models are based on the MLP or its variations and use different techniques for adapting the learning rate ϵ of Eq. (28).

A. McCLELLAND'S NEW ERROR MEASURE

In this method [18], the total error of Eq. (27) is redefined as

$$E(\mathbf{w}) = -\sum_{j,c} \ln\left[1 - \left(y_j^H(\mathbf{w}) - d_{j,c}\right)^2\right] \tag{33}$$

such that

$$\frac{\partial E}{\partial y_j} = \frac{1}{1 + \left(d_j - y_j^H\right)} - \frac{1}{1 - \left(d_j - y_j^H\right)} \tag{34}$$

for $h = H$ in Eq. (30). This error derivative is expected to speed up the movement of weights that had previously moved slowly because of small sigmoid derivatives. Hence in case of output units whose output is at the wrong end of the sigmoid (and close to 0 or 1), the weight change increases and thereby the learning time is reduced.

B. LEARNING RATE ADAPTED BY ANGLES

In this scheme proposed by Chan and Fallside [19], useful information about the shape of the energy contour can be learned from the directions of the local gradient vector $\nabla\mathbf{E}(t)$ and the weight updates $\Delta\mathbf{w}(t-1)$ and $\Delta\mathbf{w}(t)$ using the vector version of Eq. (28). The angle $\theta(t)$, giving an indication of the nature of the energy surface during training, is defined as

$$\cos\theta(t) = -\frac{\nabla\mathbf{E}(t)\cdot\Delta\mathbf{w}(t-1)}{\|\nabla\mathbf{E}(t)\|\,\|\Delta\mathbf{w}(t-1)\|}. \tag{35}$$

The learning rate is adapted as

$$\epsilon(t) = \epsilon(t-1)\left(1+\tfrac{1}{2}\cos\theta(t)\right) \tag{36}$$

such that $\epsilon(t)$ decreases near the *ravine* walls when $\pi/2 \leqslant \theta(t) \leqslant 3\pi/2$ and increases at the *plateaus* when $\theta(t)\to 0$ or 2π. The damping coefficient is modified as

$$\alpha(t) = \lambda(t)\epsilon(t),$$

where

$$\lambda(t) = \lambda(0)\frac{\|\nabla\mathbf{E}(t)\|}{\|\Delta\mathbf{w}(t-1)\|}$$

and $0 \leqslant \lambda(0) \leqslant 1$. This scheme is supposed to reduce oscillations at the walls of the ravine.

C. *ADAPTIVE ACCELERATION STRATEGY SSAB*

This is a modification of the strategy reported by Jacobs [22]. In this approach by Tollenaere [20], (i) every weight w_{ij} has its own individual (adaptive) step size ϵ_{ij}, (ii) each step size ϵ_{ij} is allowed to vary over time, (iii) for each w_{ij}, as long as the w_{ij} derivative does not change sign, the corresponding ϵ_{ij} is increased, and (iv) when a change in the sign of the w_{ij} derivative is detected, (a) the previous weight update is undone and then ignored in the momentum term of the following step and (b) the corresponding ϵ_{ij} is decreased.

Therefore, as long as the weight derivative keeps changing sign, the step size is decreased until a step can be done without causing the weight derivative to change sign. This method requires a number of local computations and is supposed to be easier to implement on parallel architecture computers. However, it also involves an increase in the total computational overhead in its attempt at increasing the speed of convergence.

D. SECOND-ORDER WEIGHT CORRECTION FOR SIGMA-PI UNITS

Considering two element conjuncts [2], the output of the jth neuron in layer h [of Eqs. (25) and (26)] is given as

$$y_j^h = f_j\Bigl(\sum_{i,k} w_{jki}^{h-1} y_k^{h-1} y_i^{h-1}\Bigr), \tag{37}$$

where $f_j(\cdot)$ is the sigmoidal function. The error derivative corresponding to the jth neuron in layer h of Eq. (29) becomes

$$\frac{\partial E}{\partial w_{jki}} = \frac{\partial E}{\partial x_j} y_i^{h-1} y_k^{h-1} \tag{38}$$

such that

$$\frac{\partial E}{\partial y_j} = \sum_{j,k} \frac{\partial E}{\partial x_k} w_{kij}^h y_j^{h+1} \quad \text{for } 0 \leqslant h \leqslant H.$$

Such nets are expected to result in better performance due to the higher order connection weight interactions. However, it should be noted that each sweep through the training set involves a much larger number of weight updates and leads to a resultant increase in the computational overhead.

E. LEARNING RATE ADAPTED BY ERROR

The total error E of Eq. (27) is used to determine the learning rate ϵ in the method reported by Silverman and Noetzel [21]. When $E(t) < E(t-1)$, ϵ is increased additively. On the other hand, when $E(t) > E(t-1)$, ϵ is decreased multiplicatively. This scheme ensures that the decrease is faster than the increase.

There is a potential pitfall in such methods if, in some dimension, the energy landscape is such that the gradient never changes [20]. In such cases the learning rate may keep growing and the weights may become infinitely large.

F. HEURISTIC SCALING OF LEARNING RATE

This technique by Franzini [18] aims to maintain a maximum value of learning rate ϵ such that the direction of weight change remains nearly constant. The angle between the error derivative vector component $d_{ij} = \partial E/\partial w_{ij}$ at cycles t and $t-1$ is defined as

$$\cos\theta = \frac{\Sigma_{i,j} d_{ij}(t-1) d_{ij}(t)}{\sqrt{\Sigma_{i,j}[d_{ij}(t-1)]^2 \Sigma_{i,j}[d_{ij}(t)]^2}}. \tag{39}$$

The epsilon-scaling rule is given as

$$\epsilon(t) = \begin{cases} \epsilon(t-1)\beta^{+}\cos\theta, & \text{if } \cos\theta > 0, \\ \epsilon(t-1)\beta^{-}, & \text{otherwise}, \end{cases} \tag{40}$$

with $\beta^{+} \simeq 1.005$ and $\beta^{-} \simeq 0.8$. This rule is supposed to significantly reduce the learning time and avoid local minima (which fixed higher values of ϵ are likely to reach).

G. *FIXED LEARNING RATE*

The conventional MLP uses fixed learning rate ϵ [2]. However, it has been pointed out in [20] that there is an *optimal step size region* (*osr*) with the interval $[\epsilon_{\text{opt}} - \delta_l, \epsilon_{\text{opt}} + \delta_r]$ for every problem. For all ϵ lying in this region, the learning converges reasonably fast and remains stable. However, one does not know a priori where the *osr* is located for a particular problem. The width of the *osr* scales with the absolute value of ϵ_{opt} [20]. The network size and training set seem to influence the *osr*.

To overcome these problems, various techniques are currently being used for heuristically adapting the learning rate ϵ. A few of these schemes are discussed in this section.

H. *MINIMIZATION OF CROSS ENTROPY*

This is a modification of the fuzzy MLP using the more standard mean square error criterion. In this approach [14], the cross-entropy S is minimized during training. We define

$$S = \sum_{j,c} \left[-d_j \ln y_j^H - (1-d_j)\ln(1-y_j^H) \right] \tag{41}$$

such that the weight updating of Eq. (28) is given as

$$\Delta w_{ji}^{h}(t) = -\epsilon \frac{\partial S}{\partial w_{ji}} + \alpha \Delta w_{ji}^{h}(t-1). \tag{42}$$

Note that

$$\frac{\partial S}{\partial w_{ji}} = \frac{\partial S}{\partial y_j} \frac{\partial y_j}{\partial x_j} \frac{\partial x_j}{\partial w_{ji}} \tag{43}$$

from Eq. (29), where

$$\frac{\partial S}{\partial y_j} = \frac{y_j^H - d_j}{y_j^H\left(1 - y_j^H\right)}$$

for $h = H$. Given a set of training cases, the likelihood of producing *exactly* the desired vectors is maximized when we minimize the cross entropy [23]. The use of cross entropy also helps to speed up the learning in cases of output units that are close to 0 when they should be close to 1 and vice versa. Note that here $\partial S/\partial x_j = y_j^H - d_j$. This technique also enables the network to attain high values of *perfect match.*

7. IMPLEMENTATION PROCEDURE: RESULTS AND COMPARATIVE STUDY

The two fuzzy neural network models [13], [14] have been used to classify three sets (A, B, C) of artificially generated linearly nonseparable pattern classes involving nonconvex decision regions. These are depicted in Figures 1–3 in the two-dimensional space F_1-F_2, each set consisting of 880 pattern points. The training set consists of the pattern vectors in the nine-dimensional (for Kohonen's net) and six-dimensional (for MLP) forms. The classification decision regarding the test set is inferred by each trained neural network. The neural models were trained on the three sets of linearly nonseparable pattern classes in succession, using different network as well as training set sizes. Two linearly nonseparable pattern classes 1 and 2 were considered in each case. The region of *no pattern points* was modeled as the class *none* (*no class*).

The effect of fuzzification at the input, by varying the radius of the π-function corresponding to the linguistic set *medium* was demonstrated for both the models. In the cases of *Pattern Sets A* and C, the decision region of class 2 was disjoint and the a priori probability of this class was much lower than that of the other classes. For this region, the contribution of the a priori probabilities in the error derivatives of the back-propagation procedure for weight updating was also investigated.

In order to demonstrate the superiority of the fuzzy neural models, an extensive comparison of their performance was made with other models. Performance of the fuzzy MLP has been compared with those of the conventional MLP, the Bayes' classifier, and seven other neural algorithms. Because the self-organizing, fuzzy Kohonen's model has been used as a classifier, its performance has been compared only with its conventional version (used as a classifier) and the Bayes' model.

A. *USING FUZZY SELF-ORGANIZATION*

Tables 1–3 are used to compare the performance on test set (both classwise and overall) of different sizes of the fuzzy self-organizing neural

TABLE 1

Comparison between recognition scores for various sizes of proposed self-organizing neural net array with $s = 0.2$, the nonfuzzy version of the model, and the Bayes' classifier, using *perc* = 50 on *Pattern Set A*.

	Proposed Fuzzy					Nonfuzzy	
Class	10 × 10	14 × 14	10 × 20	20 × 10	16 × 16	14 × 14	Bayes
1	33.0	77.4	72.1	68.2	65.6	96.9	100.0
2	44.9	67.3	79.6	87.7	69.3	0.0	20.4
None	72.2	55.5	40.7	44.4	53.7	4.3	24.0
Overall	48.8	68.3	59.0	61.7	61.2	52.1	63.2

TABLE 2

Comparison between recognition scores for various sizes of proposed self-organizing neural net array with $s = 0.2$, the nonfuzzy version of the model, and the Bayes' classifier, using different values of *perc* on *Pattern Set B*.

	Proposed Fuzzy					Nonfuzzy	Bayes	
	14 × 14		16 × 16		18 × 18	16 × 16		
Class \ *perc*	50	10	50	10	50	50	50	10
1	63.4	58.7	83.0	64.1	51.7	10.7	38.4	27.3
2	50.5	30.8	55.6	30.2	40.2	6.1	34.0	50.8
None	50.4	78.1	55.6	63.5	56.0	94.4	72.8	71.9
Overall	53.7	62.8	62.6	56.4	51.5	53.7	55.5	55.9

TABLE 3

Comparison between recognition scores for various sizes of proposed self-organizing neural net array with $s = 0.2$, the nonfuzzy version of the model, and the Bayes' classifier, using different values of *perc* on *Pattern Set C*.

	Proposed Fuzzy						Nonfuzzy	Bayes	
	10 × 10	14 × 14		16 × 16		18 × 18	16 × 16		
Class \ *perc*	10	50	10	50	10	50	50	50	10
1	43.3	77.3	50.6	82.7	51.7	72.7	53.4	100.0	79.4
2	39.1	53.8	58.7	19.2	84.7	65.3	0.0	0.0	0.0
None	81.7	44.5	51.6	55.4	35.8	53.5	72.9	0.0	28.6
Overall	56.6	64.4	51.5	69.4	48.0	65.5	57.1	58.9	57.0

net [13] on the three sets of linearly nonseparable patterns A, B, and C, respectively. Various training set sizes were used by randomly choosing *perc*% samples from each representative pattern class. The remaining $(100 - perc)\%$ samples from the original data set were used as the test set in each case. We selected $fdenom = 0.8$ in Eqs. (8) and (9), $b = 0.02$ in Eq. (14), $cdenom = 100$ and $\alpha' = 0.9$ in Eq. (16), $kn = 10$ in Eq. (19), and $\delta = 0.0001$ in Eq. (20) after several experiments.

Comparison was made with the performance of the Bayes' classifier and the nonfuzzy version of the network. The standard Bayes' classifier for multivariate normal patterns was used with the a priori probabilities $p_i = |C_i|/N$, where $|C_i|$ indicates the number of patterns in the ith class and N is the total number of pattern points. The covariance matrices were different for each pattern class. The proposed fuzzy models were found to result in better performances in all the three cases, considering individual classwise behavior. It is to be noted that here an n-dimensional feature space was decomposed into 3^n overlapping subregions corresponding to the three linguistic properties. This enabled the model to utilize more local information about the feature space [15] and was, therefore, better equipped to classify the given linearly nonseparable patterns. This is particularly evident on observing the classification efficiency of the nonfuzzy neural model for class 2 (whose recognition score is very poor) in the case of all three pattern sets.

In addition, Table 4 shows a comparison in the classification performance of the fuzzy neural model on *Pattern Set A* using different values of the scale factor s of Eq. (12) (during self-organization). Here $s = 0.2$ is found to yield the best results. It may be noted that large values of s result in a greater dependency on the contextual class information part of the input vector during self-organization. Hence during testing (when $s = 0$)

TABLE 4

Comparison between recognition scores for various values of scale factor s using 14×14 self-organizing neural net array with *perc* = 50 on *Pattern Set A*.

Class	Scale Factor s				
	0.1	0.2	0.3	0.5	1.0
1	70.0	77.4	66.0	39.1	40.0
2	69.3	67.3	73.4	73.4	93.8
None	58.0	55.5	62.3	51.2	44.4
Overall	65.5	68.3	65.5	47.4	47.6

the input vector becomes less *complete*, thereby leading to a poorer recognition score on the test set consisting of the input feature information part only. It may be noted that $0 < s < 0.5$ appears suitable in this context.

The fuzzy partitioning for the pattern classes along with the hard partitioning of the output space are also depicted for each set of patterns. Figures 4–6 illustrate the output maps generated for the three pattern sets A, B, and C, respectively, using 50% of the samples from each representative class during self-organization. In each case, parts (a) and (b) show the fuzzy partitioning for classes 1 and 2 separately (for the sake of clarity), whereas part (c) gives the hard partitioning of the output space (considering all classes). Note that along the horizontal axes each pair of numerals denotes the location of a single neuron whereas along the vertical axis one numeral corresponds to a single neuronal location. This is done to compensate for the difference in resolution along the two perpendicular axes. A comparison of these output maps with the corresponding original pattern sets in Figures 1–3 brings forth the utility of the fuzzy neural net in modeling the given linearly nonseparable pattern sets.

B. USING THE FUZZY MLP

A detailed study was made with the fuzzy MLP [14] in the *batch mode* of updating. Various numbers of hidden nodes m were used in conjunction with different training set sizes perc% (chosen as explained in Section 6A). We selected $fdenom = 0.8$ in Eqs. (8) and (9) and $kn = 10$ and $\delta = 0.0001$ in Eq. (32) after several experiments. The effect of using a different number of hidden layers was investigated. The performance of a modified version of the proposed fuzzy MLP that minimizes cross-entropy (model H) (reported in [14]) instead of the more standard mean square error measure (model O) has also been demonstrated (in the case of *Pattern Set A*).

```
1111111111111111111111111111
1111111111111111111111111111
11111111    111111    111111
111111                  1111
1111                  111111
1111                  111111
1111                  111111
1111                    1111
1111                    1111
1111                  111111
11111111            11111111
1111111111      111111111111
1111111111111111111111111111
    1111111111111111111111
```

(a)

```



              22222222
      22      22222222
    222222    22222222
    222222    2222222222
    22222222  2222222222
      222222    22222222
      222222      22




```

(b)

```
  11111111111111111111111111
1111111111111111111111111111
111111                  1111
1111                    1111
1111            222222  1111
1111  2222    22222222  1111
11  222222      222222221111
11    2222      222222221111
1111    22        2222221111
1111                    1111
111111                111111
11111111            11111111
    111111    11111111111111
    11111111111111111111
```

(c)

Fig. 4. Output map generated by 14 × 14 self-organizing neural net array with *perc* = 50 and $s = 0.2$ for *Pattern Set A*. (a) Fuzzy partitioning for class 1; (b) fuzzy partitioning for class 2; (c) hard partitioning of the output space.

Tables 5 and 6 demonstrate the effect of using three and more layers (having m nodes in each hidden layer) and different training set sizes on the proposed neural net (model O) for the three pattern sets (A, B, C). In Table 5 an additional comparison is provided with the model H for *Pattern*

```
        111111111111111111  1111
        111111111111111111  1111
                  111111    1111
                            1111
                            1111
                            1111
                            1111
              111111        1111
          1111111111111111111111
          1111111111111111111111
        111111111111111111111111
        1111111111111111111111
        1111111111111111111111
```

(a)

```


  22  22
  2222222222222222222222
222222222222222222222222
222222222222222222222222
2222        222222222222
2222
2222
2222
2222
2222

    222222        2222
222222222222222222222222
222222222222222222222222
```

(b)

```
          11111111111111    1111
          11111111111111    1111
                            1111
    222222222222  2222      1111
  2222222222222222222222    1111
222222222222222222222222    1111
2222            222222      1111
22                          1111
2222          111111      111111
22        1111111111111111111111
22        1111111111111111111111
22        111111111111111111
            11  111111  11

    222222        2222
2222222222222222222222
```

(c)

Fig. 5. Output map generated by 16×16 self-organizing neural net array with $perc = 50$ and $s = 0.2$ for *Pattern Set B*. (a) Fuzzy partitioning for class 1; (b) fuzzy partitioning for class 2; (c) hard partitioning of the output space.

```
  111111111111111111111111111111
11111111111111111111111111111111
1111111111      1111111111111111
111111              111111111111
1111                    11111111
111111                  11111111
11111111      111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
111111                  11111111
111111                  11111111
111111                1111111111
111111111111111111  111111111111
11111111111111111111111111111111
  1111111111111111111111111111
```

(a)

```
            22222222
          2222222222
          2222222222
            22222222
              2222


            22
          222222
          22222222
          22222222
            222222
```

(b)

```
  111111111111111111111111  1111
  111111111111111111111111111111
11111111      2222  111111111111
1111        22222222    11111111
1111        22222222      111111
1111          2222      11111111
111111                  111111
1111111111    111111111111111111
  11111111  11111111111111111111
1111111111            1111111111
1111        22          11111111
1111        2222        11111111
1111        2222        11111111
1111      11            1111111111
  1111111111111111111111111111
  11111111111111111111111111
```

(c)

Fig. 6. Output map generated by 16 × 16 self-organizing neural net array with *perc* = 50 and $s = 0.2$ for *Pattern Set C*. (a) Fuzzy partitioning for class 1; (b) fuzzy partitioning for class 2; (c) hard partitioning of the output space.

TABLE 5

Comparison in output performance for various sizes of the proposed neural net model, having m nodes per hidden layer and using different values of *perc*, with model H for *pattern Set A*.

	Model O								Model H			
Layers	3	3	4	4	4	4	5	5	3	4	4	4
perc	10	50	10	10	10	50	10	50	10	10	10	50
Nodes m	16	16	17	18	19	19	11	11	16	18	19	19
Best b (%)	100.	87.7	100.	100.	100.	98.	100.	95.7	98.9	98.9	100.	98.
Perfect p (%)	16.1	18.7	69.	77.	81.6	53.6	73.6	70.	66.7	98.9	95.4	84.1
mse	.008	.046	.005	.003	.002	.016	.004	.023	.016	.003	.0007	.011
Test												
mse_t	.066	.075	.112	.065	.067	.028	.086	.042	.071	.124	.06	.03
Class 1 (%)	90.5	91.7	88.1	91.3	93.2	95.6	96.1	94.3	91.8	92.	93.7	95.6
Class 2 (%)	86.3	46.9	67.	69.3	71.6	91.8	59.1	83.6	84.1	68.2	76.1	87.7
None (%)	85.2	86.4	73.5	84.2	82.4	90.7	76.3	90.7	78.7	61.5	83.5	91.3
Overall t (%)	88.1	84.8	80.4	86.2	86.9	93.4	84.7	91.8	86.1	78.2	88.	93.2

TABLE 6

Output performance for various sizes of the proposed neural net model, having m nodes per hidden layer and using different values of *perc*, on *Pattern Sets B* and *C*.

	Pattern Set B						*Pattern Set C*					
Layers	3	3	4	4	5	5	3	3	4	4	5	5
perc	10	50	10	50	10	50	10	50	10	50	10	50
Nodes m	11	11	14	14	14	14	13	13	12	12	13	13
Best b (%)	100.	86.6	100.	99.3	100.	99.6	100.	67.9	100.	97.3	100.	96.6
Perfect p (%)	62.1	12.1	65.5	69.1	87.4	80.	32.2	0.	73.6	37.6	79.3	60.4
mse	0.007	0.086	0.004	0.008	0.005	0.005	0.008	0.159	0.003	0.024	0.005	0.026
Test												
mst_t	0.088	0.09	0.09	0.041	0.095	0.037	0.143	0.168	0.162	0.042	0.097	0.036
Class 1 (%)	78.6	93.7	88.5	96.4	87.	97.3	83.9	100.	79.7	95.7	82.	95.
Class 2 (%)	84.	68.	77.7	91.7	89.7	94.8	84.8	0.	28.2	84.6	71.7	92.3
None (%)	84.9	84.9	83.9	88.3	78.6	91.8	59.5	0.	62.7	87.1	81.	89.6
Overall t (%)	83.1	83.4	83.7	91.1	83.2	93.8	75.4	58.9	70.7	92.	81.1	92.9

Set A. The *perfect match p*, *best match b*, and *mean square error mse* correspond to the training set, whereas the remaining measures refer to the test set. Table 5 demonstrates using *Pattern Set A* that model H always converged to an appreciably better *perfect match* as compared to the corresponding version of model O.

C. FUZZIFICATION AT THE INPUT

The effect of varying the amount of fuzziness of the input vector was investigated for both fuzzy models [13], [14]. The input feature information is given in the $3n$-dimensional space of Eq. (4) in terms of the linguistic property sets *low*, *medium*, and *high*. The π-functions that represent these properties are defined by the radius λ and center c values given by Eqs. (7)–(9).

Varying λ_{medium} while keeping λ_{low} and λ_{high} fixed [by Eqs. (8) and (9)], one can alter the overlapping among the three π-functions. Let $\lambda_{medium} = fnos * \lambda_{medium}$, where $fnos = 1$ for the value of λ_{medium} given by Eq. (7). As we decrease *fnos*, the radius λ_{medium} decreases around c_{medium} such that ultimately there is insignificant overlapping between the π-functions *medium* and *low* or *medium* and *high*. This implies that certain regions along the feature axis F_j go underrepresented such that $\mu_{low(F_{ij})}(\mathbf{X}_i)$, $\mu_{medium(F_{ij})}(\mathbf{X}_i)$, and $\mu_{high(F_{ij})}(\mathbf{X}_i)$ attain small values. Note that the particular choice of values of the λs and cs by Eqs. (7)–(9) ensure that for any pattern point $\mathbf{X}_i$ along F_j, at least one of $\mu_{low(F_{ij})}(\mathbf{X}_i)$, $\mu_{medium(F_{ij})}(\mathbf{X}_i)$, and $\mu_{high(F_{ij})}(\mathbf{X}_i)$ is greater than 0.5 [14]. On the other hand, as we increase *fnos* the radius λ_{medium} increases around c_{medium} such that the amount of overlapping between the π-functions increases.

Tables 7 and 8 demonstrate the performance of the fuzzy Kohonen's net and the fuzzy MLP, respectively, with different values of *fnos* on *Pattern Set A*. Kohonen's model was implemented on a 14×14 array using $perc = 50$ whereas the MLP used one hidden layer that had 17 nodes with $perc = 10$. This was done to maintain uniformity with the results of Tables 1 and 5, respectively. It was observed that for the Kohonen's model $0.8 < fnos < 1.2$ gave good results whereas the MLP generated good performance for $0.7 < fnos < 1.2$. This implies that the amount of overlapping signified by Eqs. (7)–(9) [14] indicated the most suitable choice of the

TABLE 7

Effect of varying fuzziness of input features for fuzzy Kohonen's model using 14×14 array with $perc = 50$ on *Pattern Set A*.

	fnos							
Class	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3
1	67.8	70.0	67.8	80.8	77.4	60.0	58.7	43.9
2	55.1	53.0	59.1	77.5	67.3	63.2	81.6	91.8
None	45.6	42.6	53.7	53.7	55.5	58.0	53.7	58.0
Overall t	58.2	58.0	61.6	70.5	68.3	59.6	59.4	54.4

TABLE 8

Effect of varying fuzziness of input features for three-layered fuzzy MLP model using $m = 17$ hidden nodes with *perc* = 10 on *Pattern Set A*.

	fnos									
	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
Best b (%)	94.3	98.9	100.	100.	100.	100.	100.	100.	100.	100.
Perfect p (%)	21.9	0.0	54.1	18.4	34.5	36.8	17.3	46.0	21.9	19.6
mse	0.039	0.019	0.006	0.008	0.006	0.006	0.007	0.005	0.008	0.012
Test										
mse_t	0.113	0.092	0.068	0.073	0.076	0.077	0.099	0.094	0.128	0.131
Class 1 (%)	85.0	90.5	90.8	88.4	89.3	86.7	87.9	86.0	78.9	86.4
Class 2 (%)	75.0	69.3	60.2	67.0	75.0	73.8	57.9	64.7	57.9	54.5
None (%)	69.7	75.9	89.0	86.6	84.5	91.0	83.8	81.7	78.3	69.4
Overall t (%)	78.3	82.8	86.7	85.3	86.0	86.8	83.1	82.1	76.4	76.6

values of the λs and cs. Very large or very small amounts of overlapping among the linguistic properties of the input feature are found to be undesirable.

D. COMPARISON OF OTHER NEURAL ALGORITHMS

A comparison was made among several layered neural net models (variations of MLP) mentioned in the following text, using the same number of hidden nodes as well as the *same set* of initial random weights as in the corresponding proposed fuzzy model O [14]. This enabled a more appropriate comparison in performance of the various models. It was observed that the overall performance of the fuzzy model was better, especially considering the fewer number of sweeps (through the training set) required in the process of convergence. The other models compared here use (i) McClelland's new error measure reported in [18] (model M), (ii) learning rate adapted by angles [19] (model C), (iii) adaptive acceleration strategy SSAB [20] (model A), which is a modification over the strategy reported in [22], (iv) second-order weight correction for sigma-pi units [2] (model S), (v) learning rate adapted by error [21] (model P), (vi) heuristic scaling of learning rate [18] (model F), and (vii) the conventional fixed learning rate [2] (model R).

The above-mentioned models were modified to incorporate fuzzy linguistic values in the $3n$-dimensional input space of Eq. (4) to facilitate a more authentic comparison with the proposed fuzzy model O. The objective was mainly to demonstrate the utility of the heuristic adaptation of the learning rate ϵ by Eq. (32). However, model M (using a different error measure) and model S (using a different network architecture) were compared incorporating the same scheme (as model O) for adapting ϵ. The performance of the traditional nonfuzzy version of the MLP (model O′), using the same learning rate variation scheme as model O, was also studied.

Tables 9–11 compare the performance of the fuzzy neural network on the three pattern sets (A, B, C) with the various other models previously

TABLE 9

Comparison between output performance of various three-layered neural models on *Pattern Set A* using $m = 17$ hidden nodes with *perc* = 10.

	Model												
		R											
	O	$\epsilon = 2.0$	$\epsilon = 1.0$	$\epsilon = 0.5$	$\epsilon = 0.3$	$\epsilon = 0.1$	P	C	M	F	A	S	O′
Best b (%)	100.	70.2	100.	98.9	96.6	78.2	100.	100.	100.	50.6	77.	93.1	73.6
Perfect p (%)	34.5	0.	9.2	2.3	1.2	10.4	4.6	15.	5.8	0.	11.5	11.5	3.5
mse	0.006	0.064	0.013	0.015	0.023	0.081	0.023	0.018	0.008	0.162	0.109	0.06	0.096
Test													
mse_t	0.076	0.13	0.084	0.087	0.087	0.132	0.09	0.088	0.119	0.161	0.169	0.12	0.119
Class 1 (%)	89.3	94.7	86.4	89.1	90.3	84.3	95.9	88.6	86.2	99.	53.8	87.2	97.8
Class 2 (%)	75.	4.5	76.1	81.8	45.4	31.8	76.1	64.7	50.	7.9	21.6	82.9	0.0
None (%)	84.5	78.7	81.1	77.3	84.2	69.4	70.1	81.4	76.3	46.	79.7	64.2	72.1
Net t (%)	86.	78.8	83.3	84.	83.1	73.	84.2	83.3	78.5	69.5	59.7	78.3	77.5

TABLE 10

Comparison between output performance of various three-layered neural models on *Pattern Set B* using $m = 11$ hidden nodes with *perc* = 10.

	Model							
	O	P	C	M	F	A	S	O′
Best b (%)	100.	97.7	94.3	100.	61.	60.9	100.	87.4
Perfect p (%)	62.1	35.7	18.4	28.8	0.	2.3	47.2	1.2
mse	0.007	0.018	0.039	0.007	0.133	0.113	0.007	0.078
Test								
mse_t	0.088	0.105	0.121	0.099	0.174	0.142	0.076	0.152
Class 1 (%)	78.6	83.1	85.	83.6	55.7	65.6	82.1	47.7
Class 2 (%)	84.	78.8	64.5	74.3	36.	32.	89.7	72.0
None (%)	84.9	81.	79.3	84.4	78.1	88.9	84.6	82.0
Overall t (%)	83.1	81.1	77.5	81.9	63.1	70.5	85.1	71.1

TABLE 11

Comparison between output performance of various three-layered neural models on *Pattern Set C* using $m = 13$ hidden nodes with $perc = 10$.

	Model							
	O	P	C	M	F	A	S	O′
Best b (%)	100.	93.1	93.1	100.	70.1	82.8	100.	77.1
Perfect p (%)	32.2	19.6	0.	33.4	10.4	9.2	10.4	8.1
mse	0.008	0.049	0.041	0.009	0.116	0.096	0.011	0.104
Test								
mse_t	0.143	0.139	0.14	0.149	0.162	0.163	0.131	0.16
Class 1 (%)	83.9	85.4	84.8	79.7	85.	92.9	90.1	87.1
Class 2 (%)	84.8	0.	0.	91.3	0.	0.	67.4	0.
None (%)	59.5	70.9	69.9	61.6	50.1	27.9	45.8	51.6
Overall t (%)	75.4	75.4	74.6	74.	67.8	64.7	73.2	69.6

mentioned, choosing m nodes in the single hidden layer and 10% of training samples from each representative class. The choice of m was made after several runs with different number of hidden nodes. Best results were obtained with $m = 17$, 11, and 13 for pattern sets A, B, and C, respectively. In all three cases the nonfuzzy model O′ gave poorer results. This is particularly evident on observing the very poor recognition score for class 2 in the case of *Pattern Sets A* and *C*. It may be noted that the use of fuzzy linguistic inputs caused the n-dimensional feature space (as in model O′) to be decomposed into 3^n overlapping subregions corresponding to the three primary properties. As mentioned before, this enabled the fuzzy model to utilize more local information about the feature space and was, therefore, better equipped to handle the linearly nonseparable pattern classes that have concave decision regions.

All models [except M and S that terminated by the criterion proposed in Eq. (32)] were run for the same number of sweeps as required by the corresponding model O before convergence. This allowed us to assess the status of the other models at the time when model O converged after having started from the same set of initial connection weights and then having been trained with the same set of pattern points. In this connection, it is worth mentioning that the recognition score depends on the terminating point of the algorithm as well as on the heuristic used for learning rate variation. However, when a neural algorithm converges to a reasonably good solution over a smaller number of sweeps through the training set, this is indicative of its efficiency. Hence, the comparison provided with the other algorithms in Tables 9–11 (using the same terminating point and uniform input representation), with different heuristics for adapting the learning rates, helps to bring out the utility of the scheme proposed in [14] for model O. Further, it is to be noted that model S involves a larger number of connection weights (second-order connections) as compared to all other models used here. Therefore, a certain number of sweeps in case of model S entail a much larger number of weight updates (increased computational overhead) as compared to the other models. Note that investigations regarding model R have been reported in detail with respect to *Pattern Set A* only (in Table 9 and Figure 8) because it uses a constant learning rate that is very much problem dependent.

On the whole, the performance of model O over *Pattern Set C* was observed to be poorer than that of its performance over the other two pattern sets. This is perhaps an indicator of the *more difficult* nature of the problem in the case of *Pattern Set C*. Comparing the results from Tables 9–11, we conclude that model O gave consistently good performances over all three of the linearly nonseparable pattern sets. On the other hand, the behavior of the other models varied over the same three cases.

Figures 7–10 illustrate the variation of the mean square error of the various layered neural net models during training with the number of sweeps, using $perc = 10$, over the three linearly nonseparable pattern sets (A, B, C). All models (except S) were assessed by their status up to the sweep number when the corresponding three-layered model O converged. Figure 7 shows the results of using variations of three-layered nets with $m = 17$ over *Pattern Set A*. We observe that in the case of model S (e) the *mse* decreased very rapidly in the initial stages, but stabilized to 0.06 after around 60 sweeps. On the other hand, models O (a), C (c), and P (d) exhibited similar behavior in the early stages with the *mse* of model O falling more rapidly after around 125 sweeps to an ultimate low value of 0.006. Model S terminated at around 190 sweeps, as indicated by the arrow along the abscissa. Note that this corresponds to a larger number of weight updates as compared to the corresponding stage in the other models. The *mse* for model M (b) decreased rapidly from an initial high value to a satisfactory final value that was lower than those obtained by models C and P, which incidentally behaved better initially. Oscillations were evident for models P and A (f) in the process of convergence. Models A (f) and F (g) behaved rather poorly.

Figure 8 demonstrates the results of using three-layered models R with $\epsilon = 2$ (f), 1 (c), 0.5 (d), 0.3 (e), 0.1 (g), and model O (b) with $m = 17$ nodes on *Pattern Set A*. It is seen that model O (b) had the best overall behavior, although (c), (d), and (e) also exhibited satisfactory performance. However, a four-layered version of model O (a) with $m = 19$ converged to a lower final value of *mse* over a fewer number of sweeps through the training set.

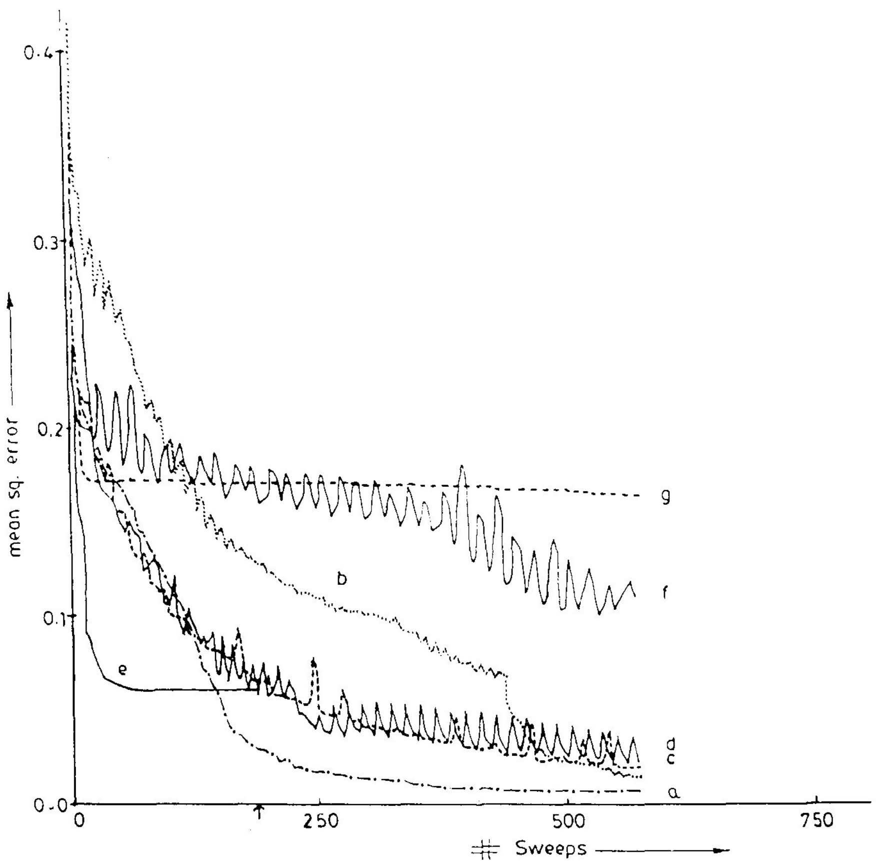

Fig. 7. Comparison of the variation of mean square error with the number of sweeps for the various three-layered neural net models (using $m = 17$ and $perc = 10$) over *Pattern Set A*. (a) Model O; (b) model M; (c) model C; (d) model P; (e) model S; (f) model A; (g) model F.

However this entailed a larger number of weight updates. The results of Figures 7 and 8 may be compared with those of Table 9 for a better understanding of the behavior of the various neural models. Note that as reported in [20], the choice of the appropriate value of ϵ is very much problem dependent. Here lies the advantage of choosing adaptive algorithms for varying the learning rate.

In Figure 9 we depict the results of using variations of the layered neural network models on *Pattern Set B*. The four-layered version of model

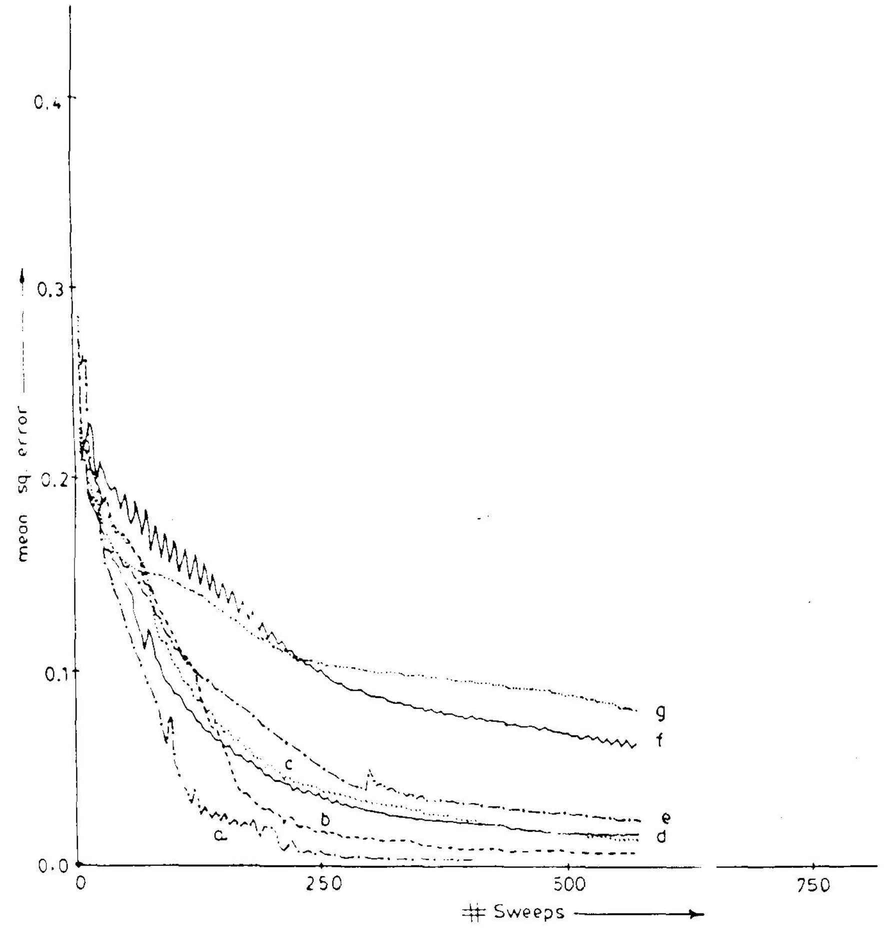

Fig. 8. Comparison of the variation of mean square error with the number of sweeps with *perc* = 10 over *Pattern Set A* between three-layered ($m = 17$ hidden nodes) models R with $\epsilon = 1.0$ (c), 0.5 (d), 0.3 (e), 0.2 (f), 0.1 (g) and (b) model O. A four-layered version of O with $m = 19$ is given (a).

O (a) with $m = 14$ did not give superior results as compared to its three-layered counterpart (c) with $m = 11$. This may also be verified from Table 6. All other models used three layers with $m = 11$ hidden nodes. The *mse* for model S(b) initially decreased rather rapidly and finally stabilized to a low value at around 100 sweeps. The arrow along the abscissa marks the point of termination of this algorithm. Model P (d) had a satisfactory

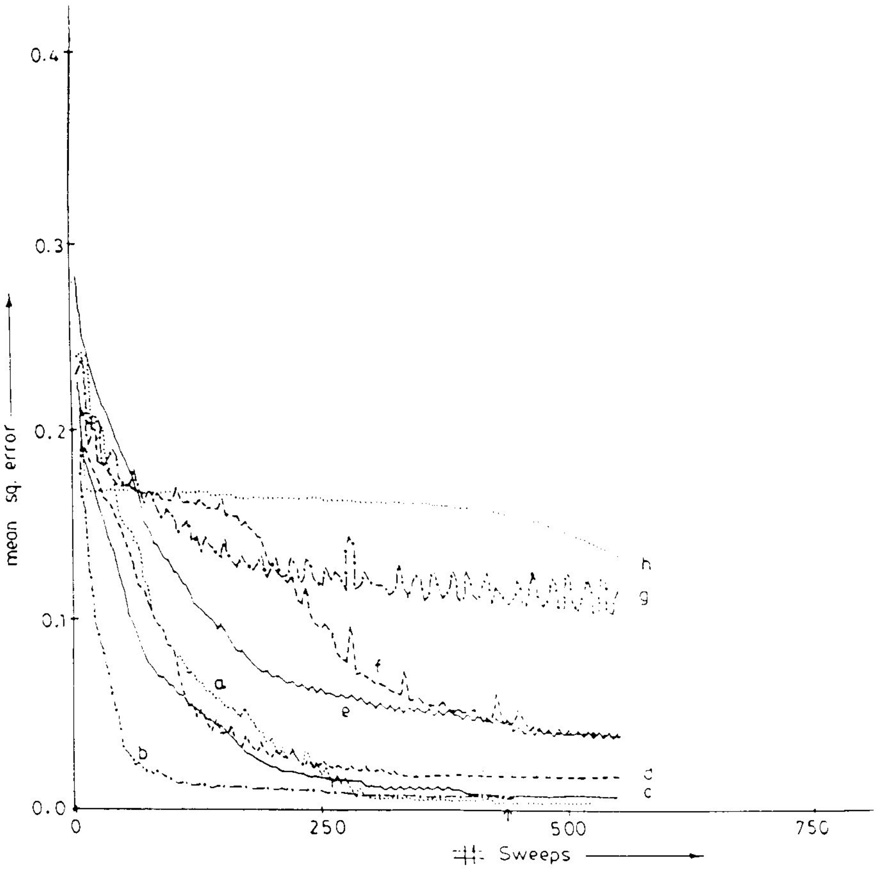

Fig. 9. Comparison of the variation of mean square error with the number of sweeps for various layered neural net models with *perc* = 10 over *Pattern Set B*. (a) Four-layered version of model O with $m = 14$. Three-layared networks with $m = 11$ for (b) model S, (c) model O, (d) model P, (e) model M, (f) model C, (g) model A, and (h) model F.

overall performance. However, models C (f) and M (e) resulted in a final larger value of *mse* at termination. Oscillations were evident for model A (g) whereas model F (h) fared the worst. These results may be compared with those of Table 10.

Figure 10 illustrates the results of using different layered neural network variations on *Pattern Set C*. The four-layered version of model O (a) with $m = 12$ gave the lowest final value of *mse*. All other models used three layers with $m = 13$ hidden nodes. The three-layered version of model O (b) had a good overall performance. Models P (e), C (d), and M (c) had

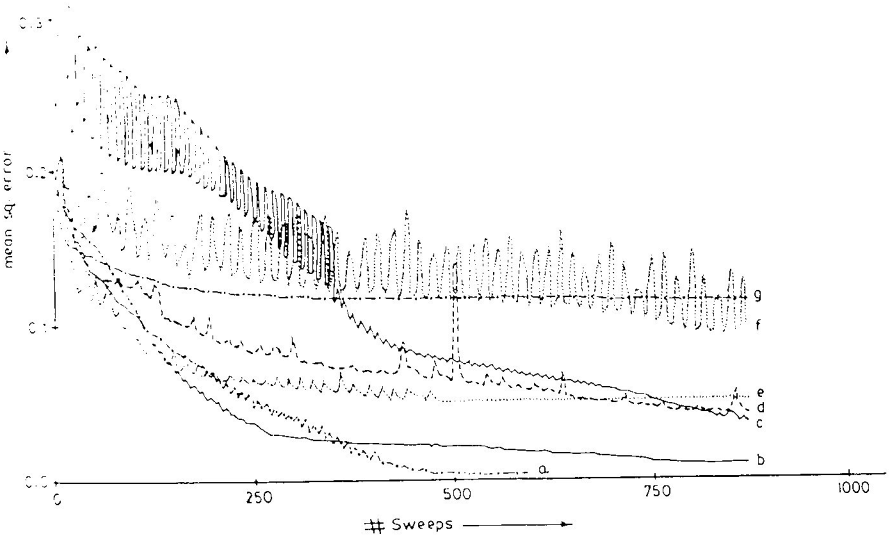

Fig. 10. Comparison of the variation of mean square error with the number of sweeps for various layered neural net models with *perc* = 10 over *pattern Set C*. (a) Four-layered version of model O with $m = 12$. Three-layered networks with $m = 13$ for (b) model O, (c) model M, (d) model C, (e) model P, (f) model A, and (g) model F.

similar final *mse* values (at termination) that were higher than that of model O. Models M (c) and A (f) exhibited a lot of oscillations in the process of convergence. The *mse* value for model M was initially rather high, but finally decreased somewhat after around 350 sweeps. Both models A (f) and F (g) behaved rather poorly. All results of this figure may be compared with those of Table 11.

E. CONTRIBUTION OF A PRIORI CLASS INFORMATION FOR BACK-PROPAGATION

It may be noted from Figures 1 and 3 that the a priori probability for class 2 in the case of *Pattern Sets A* and *C* is very low as compared to that of the other two classes. Therefore, the contribution of class 2 pattern vectors toward weight correction of the MLP (positioning of the decision surface) by Eqs. (28)–(30) is much smaller relative to the contribution of the other cases. This makes the nonfuzzy model O′, with its n-dimensional input space, unable to recognize test patterns from class 2 in case of *Pattern Sets A* and *C* (as observed from Tables 9 and 11).

In order to take into account this fact, an error correction term is introduced in Eq. (30). The modified equation becomes

$$\frac{\partial E}{\partial y_j} = \left(y_j^H - d_j\right) * l(1 - p_j), \quad \text{for } h = H, \tag{44}$$

where $p_j = |C_j|/N$ is the a priori probability of class C_j and l indicates the number of pattern classes. The correction term ensures that the lower the value of p_j, the higher is its contribution in positioning the decision surface.

Table 12 demonstrates the effect of Eq. (44) to the back-propagation procedure of the MLP in improving the performance of the model in the case of *Pattern Sets A* and *C*. Note that this modification is effective in the case of pattern classes that have widely varying a priori probabilities. The use of the multiplicative factor serves to counteract the insignificant

TABLE 12
Comparison between the output performance of three-layered nonfuzzy (O′) and fuzzy (O) neural models with their corresponding extensions (using a priori probability contributions) O'_p and O_p on *Pattern Sets A* and *C*, using m hidden nodes and $perc = 10$.

	Pattern Set A ($m = 17$)				*Pattern Set C* ($m = 13$)			
	O′	O'_p	O	O_p	O′	O'_p	O	O_p
Best b (%)	73.6	72.5	100.	100.	77.1	86.2	100.	100.
Perfect p (%)	3.5	0.0	34.5	1.2	8.1	5.8	32.2	41.4
mse	0.096	0.094	0.006	0.008	0.104	0.06	0.008	0.006
Test								
mse_t	0.119	0.125	0.076	0.08	0.16	0.145	0.143	0.135
Class 1 (%)	97.8	95.9	89.3	87.4	87.1	89.1	83.9	80.9
Class 2 (%)	0.0	45.4	75.0	73.8	0.0	40.9	84.8	76.0
None (%)	72.1	57.0	84.5	88.3	51.6	59.8	59.5	69.9
Overall t (%)	77.5	76.0	86.0	86.2	69.6	73.0	75.4	76.8

contribution (to weight correction and hence to the positioning of the decision surface) of a pattern class that has very few samples.

Models O'_p and O_p refer, respectively, to the variations of the three-layered models O′ (nonfuzzy) and O (fuzzy) using the contribution of the a prior probabilities. Each network used m hidden nodes with $perc = 10$ training samples chosen from each pattern class. Note that the recognition score of class 2 patterns improved radically for both *Pattern Sets A* and *C* in case of model O'_p. The performance on the whole was superior in the case of both models O'_p and O_p (relative to models O′ and O, respectively) in the case of *Pattern Set C* (involving more complicated decision regions). The relative improvement was always more noticeable in the case of model O'_p (the nonfuzzy version).

8. CONCLUSIONS AND DISCUSSION

The effectiveness of using fuzzy versions of the Kohonen's net [13] and the MLP [14] in classifying certain linearly nonseparable pattern classes with nonconvex decision regions has been demonstrated. It may be noted that such patterns cannot be properly classified by the Bayes' classifier for normal distributions or other metric-based methods. The components of the input vector of these fuzzy models [13], [14] consist of the membership values to the overlapping partitions of linguistic properties *low*, *medium*, and *high* corresponding to each input feature. The use of linguistic inputs enables an n-dimensional feature space to be decomposed in 3^n overlapping subregions such that more local information of the feature space can be used. On the other hand, the neural models help in generating the required concave and/or disconnected decision regions. The fusion of these concepts, therefore, enabled the fuzzy neural models to provide a superior performance in classifying the given linearly nonseparable pattern sets of Figures 1–3. The effect of fuzzification at the input, by varying the radius of the π-function corresponding to the linguistic set *medium*, was demonstrated for both models. The contribution of the a priori probabilities of the pattern classes in the back-propagation procedure for weight updating was seen to be effective in classifying patterns from classes with a low a priori probabilities.

Performance of the fuzzy MLP was compared with those of the conventional MLP, the Bayes' classifier, and seven other neural algorithms. Because the self-organizing fuzzy Kohonen's model was used as a classifier, its performance was compared only with its conventional version (used as a classifier) and the Bayes' model. Similar work has also been done on Indian Telugu vowel data, but has not been reported here due to space limitations. It may be noted that the fuzzy self-organizing model used partial supervision, with $s > 0$, during self-organization. On the other hand, the fuzzy MLP was fully supervised during training. This accounts for the comparatively better performance of the fuzzy MLP as compared to that of the fuzzy self-organizing network (with both models being used as classifiers). This may also be verified by comparing Tables 5 and 6 (fuzzy MLP) with Tables 1–3 (fuzzy self-organizing network).

REFERENCES

1. R. P. Lippmann, An introduction to computing with neural nets. *IEEE Acoustics, Speech and Signal Process.* 61:4–22 (1987).
2. D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing*, Vol. 1. MIT, Cambridge, MA, 1986.
3. T. Kohonen, *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1989.
4. G. J. Klir and T. Folger, *Fuzzy Sets, Uncertainty and Information*. Addison-Wesley, Reading, MA, 1989.
5. L. A. Zadeh, Making computers think like people. *IEEE Spectrum* August: 26–32 (1984).
6. J. C. Bezdek and S. K. Pal, eds., *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data*. IEEE Press, New York, 1992.
7. A. Kandel, *Fuzzy Mathematical Techniques with Applications*. Addison-Wesley, Reading, MA, 1986.
8. H.-J. Zimmermann, *Fuzzy Set Theory and its Applications*. Kluwer, Boston, 1991.
9. *Proceedings of the Second International Conference on Fuzzy Logic and Neural Networks* (*IIZUKA*92), Iizuka, Fukuoka, Japan, July 1992.
10. *Proceedings of the First IEEE International Conference on Fuzzy Systems* (*FUZZ-IEEE*), San Diego, March 1992.
11. Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, Reading, MA, 1989.
12. B. Kosko, *Neural Networks and Fuzzy Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1991.
13. S. Mitra and S. K. Pal, Self-organizing neural network as a fuzzy classifier. *IEEE Trans. Syst., Man and Cybernet.* To appear.
14. S. K. Pal and S. Mitra, Multi-layer perceptron, fuzzy sets and classification. *IEEE Trans. Neural Networks* 3:683–697 (1992).
15. S. K. Pal and D. P. Mandal, Linguistic recognition system based on approximate reasoning. *Inform. Sci.* 61:135–161 (1992).
16. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.
17. M. Minsky and S. Papert, *Perceptrons*. MIT Press, Cambridge, MA, 1969.
18. M. A. Franzini, Speech recognition with back propagation. *Proceedings of 9th Annual Conference of the Engineering in Medicine and Biology Society*, 1702–1703. IEEE, New York, 1987.

19. L. W. Chan and F. Fallside. An adaptive training algorithm for back propagation networks. *Comput. Speech and Language* 2:205–218 (1987).
20. T. Tollenaere. Super SAB: fast adaptive back propagation with good scaling properties. *Neural Networks* 3:561–573 (1990).
21. R. H. Silverman and A. S. Noetzel. Image processing and pattern recognition in ultrasonograms by backpropagation. *Neural Networks* 3:593–603 (1990).
22. R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks* 1:295–307 (1988).
23. G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence* 40:185–234 (1989).