Selecting Useful Groups of Features in a Connectionist Framework

Debrup Chakraborty and Nikhil R. Pal, *Fellow, IEEE*

***Abstract*—Suppose for a given classification or function approximation (FA) problem data are collected using l sensors. From the output of the ith sensor, n_i features are extracted, thereby generating $p = \sum_{i=1}^{l} n_i$ features, so for the task we have $X \subset \Re^p$ as input data along with their corresponding outputs or class labels $Y \subset \Re^c$. Here, we propose two connectionist schemes that can simultaneously select the useful *sensors* and learn the relation between X and Y. One scheme is based on the radial basis function (RBF) network and the other uses the multilayered perceptron (MLP) network. Both schemes are shown to possess the universal approximation property. Simulations show that the methods can detect the bad/derogatory groups of features *online* and can eliminate the effect of these bad features while doing the FA or classification task.**

***Index Terms*—Classification, feature selection, multilayered perceptron networks, radial basis function (RBF) networks.**

I. Introduction

IT IS known that for a given problem all features that characterize a data point may not be equally important; some features may even have *unfavorable* influence on the task at hand. Feature selection techniques aim to discard the bad/irrelevant features from the available set of features. This reduction may improve the performance of classification, function approximation, and other pattern recognition systems in terms of speed, accuracy, and simplicity. We want to emphasize that the utility/suitablity of features depends on the machine learning tool being used and the problem being solved. For an easy-to-classify data set, there may exist a set of features that would be equally good with different machine learning tools, but usually this is not the case. If there exists a feature that is necessary for discrimination, then every feature selection method should select it. On the other hand, a given data set may have many correlated features and there may be different subsets of features that are equally good for the task at hand using a given machine learning tool. If features are ranked looking at the properties of each feature and/or ignoring the tool that will be finally used to design the pattern recognition system, then one may end up with a set of features with too much of redundancy and may not be able to exploit the dependency of utility of features on the tools used.

Manuscript received August 21, 2006; revised February 16, 2007 and June 26, 2007; accepted July 12, 2007.

D. Chakraborty is with the Department of Computer Science, CINVESTAV-IPN, Mexico D.F. 07360, Mexico (e-mail: debrup@cs.cinvestav.mx).

N. R. Pal is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700108, India (e-mail: nikhil@isical.ac.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.910730

The problem of feature selection has been well addressed in literature and it has been tried out in various paradigms. Previous studies on feature subset selection focused mainly around statistical approaches like principle component analysis (PCA) [23], linear discriminant analysis (LDA) [14], etc. These methods attempt to reduce the dimensionality of the feature space by creating new features which are combination of the original ones. Hence, strictly speaking, PCA and related methods are *feature extraction* techniques which extract a new set of features from the available set of features, and the dimensionality of the extracted feature space is less than that of the original one. The main drawback of these methods is that the new features lose their original identity. Leaving aside the classical PCA and LDA techniques, there have also been many other works on feature selection in the statistical framework; some of them are [22], [24], and [37]. In [25], a unified framework to compare different feature selection algorithms with different objectives is proposed.

Blum and Langley [7] have given an excellent survey on feature selection in machine learning. These approaches are different in evaluation of the feature subsets. One can broadly classify the approaches as filter and wrapper approaches. In filter approach, the feature evaluation index is independent of the main classification/function approximation algorithm, whereas in wrapper approaches the features are evaluated by the main algorithm itself. Wrapper approaches are considered better as the relevance of a feature is generally dependent on the task being performed and also on the tool being used to do the task [27].

There are many feature selection algorithms that use soft computing/computational intelligence tools. Methods described in [8] and [41] use genetic algorithms to select the relevant feature subsets. Methods described in [3], [13], [43]–[45], and [49] and a variety of others use neural networks for feature selection. Feature selection has also been attempted using fuzzy and neurofuzzy techniques [12], [42]. There are also specialized methods to deal with feature selection for very large dimensional data sets that are typical in application areas such as bioinformatics [1], [32].

In [33]–[35], MacKay has considered neural network learning in a Bayesian framework. MacKay and Neal proposed a feature selection mechanism in the Bayesian learning framework called automatic relevance detection (ARD) [36]. In the ARD model, each input variable is associated with a hyperparameter that controls the magnitude of the weights of connections out of that input unit. The significance of an input variable is determined according to the posterior distributions of these hyperparameters.

In [39], Pal and Chintalpudi developed an integrated feature selection and classification scheme based on the multilayer perceptron (MLP) architecture. The feature selection phase in their

method was integrated with the main learning task, and the MLP learned certain feature modulators along with the conventional weights and biases of a neural network. In [9], a neurofuzzy system was developed for simultaneous feature selection and system identification. The methodology developed in [9] was modified for a classifier in [10].

We term the feature selection methods described in [9], [10], and [39] and the method described in this paper as online methods. They are in spirit with the wrapper methods of feature selection. In wrapper methods, although the feature evaluation is done through the classification/function approximation algorithms, in most methods, one uses a fixed classifier or function approximation system, and the best subset from the available set of features is chosen by evaluating each possible subset or by a suitable search technique. However, in an online method, the feature selection phase is integrated with the task of learning other parameters of the system. There are other methods available in literature which can also be classified as online. The ARD [36] and its variant [46] also learn hyperparameters associated with the input features. Some evolutionary techniques [47] also learn the importance of input features along with other parameters of the classifier. For the classification problem, LIKNON [5] uses a linear programming formulation to learn feature weights along with other parameters of the separating hyperplane.

This paper addresses the problem of feature selection in a different setting. Here, we assume that the features available can be divided into a few groups. The motivation of the problem comes from the fact that today for a given problem we often obtain data from multiple sensors. For example, in an intelligent welding inspection system, the sensors could be radiograph, acoustic emission, thermograph, eddy-current detector, etc. The sensory information obtained from various sensors in the raw form may not always be useful. Hence, from a single sensory information, one may generate/extract several features. If we use all these sensors, then the design cost and complexity of the hardware will be more. Moreover, the learning task will also become more difficult. Consequently, the designer tries to reduce the number of sensors without hampering the system's performance, so the problem is the selection of useful sensors where each sensor generates a set of features. Conventional feature selection methods select good features from all available features generated from all these sensors. However, our objective here is to discard all features obtained from bad *sensors*, if any. In other words, we aim to discard sensors which are not *necessary* for a given problem.

This problem is different from the feature selection problem. We call this group feature selection (GFS). Sensor selection is a special type of GFS, where each feature group corresponds to a sensor. This kind of grouping results in a natural partition of the total set of features according to their sensory origin. In this case, selecting good feature groups is equivalent to the selection of good (relevant) sensors. Such GFS can thus help to discard poor sensors which can, consequently, yield systems with low hardware and computational costs. Sometimes, it can reduce the time to make decisions, which is very important for many applications including medical applications. There may exist other natural groupings among features as well. For example, given an image, there could be features based on cooccurance matrix [15] and wavelet analysis. In this case, the set of cooccurance-based features can form one group while the wavelet-based features can give another group. The selection of individual feature is a special case of this GFS methodology. *To our knowledge, this problem has not been addressed in literature*. We have reported some preliminary results of this investigation in [11]. Note that the issue of how to group the available set of features is not within the scope of this work. We assume that the available features can be grouped in some natural way, for example, according to their sensory origin.

We use two connectionist schemes to deal with the problem of GFS. The first scheme is a modified radial basis function (RBF) network which we call group feature selecting radial basis function (GFSRBF) network and the other is a modified MLP called the group feature selecting multilayered perceptron (GFSMLP). In both methods, the user needs to specify the groupings that exist between the features. The networks are designed to discard the effect of the bad groups. The basic philosophy of both schemes are highly inspired by [39], but both schemes are quite different from that in [39] as our schemes have the ability to select groups of useful features and thus, as discussed earlier, can be applied to the task of sensor selection. Moreover, in [39], the authors discussed only a model for the MLP framework. The GFSMLP network may be viewed as a generalization of the method in [39].

The rest of this paper is organized as follows. In Section II, we discuss the GFSRBF network along with suitable learning rules. Then, in Section III, we discuss the GFSMLP. Finally, in Section IV, we present results on some well-known classification and function approximation problems. In Section VI, the paper is concluded. In the Appendix, we discuss the universal approximation properties of GFSRBF and GFSMLP.

II. GFSRBF Network

Given an input data set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \Re^p$, an RBF network computes the function

$$\boldsymbol{F}^*(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{w}_i \phi_i(\boldsymbol{x})$$

where the ϕ_i's are the basis functions. If we assume Gaussian-type basis functions, then

$$\phi_i(\boldsymbol{x}) = \exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2}{\sigma_i^2}\right\}. \tag{1}$$

In (1), $\boldsymbol{\mu}_i$ and σ_i are the parameters related to the ith basis function, commonly known as the center and spread, respectively, and $\|.\|$ is the Euclidean norm. Let us assume $\boldsymbol{x} = [x_1\, x_2 \ldots x_p]^T$ and $\boldsymbol{\mu}_i = [\mu_{i1}\, \mu_{i2} \ldots \mu_{ip}]^T$. Then, we have

$$\phi_i(\boldsymbol{x}) = \prod_{j=1}^{p} \exp\left\{-\frac{(x_j - \mu_{ij})^2}{\sigma_i^2}\right\}. \tag{2}$$

We assume that our data are generated by l sensors and, from the sensor i, we generate n_i $(i = 1, 2, \ldots, l)$ features, so $\sum_{i=1}^{l} n_i = p$. Let the features from each sensor i be denoted by a vector $\boldsymbol{s}^i$. Hence, we can write $\boldsymbol{x} = [\boldsymbol{s}^1\, \boldsymbol{s}^2 \ldots \boldsymbol{s}^l]^T$.

Similarly, we can write the vector representing the center as $\boldsymbol{\mu}_i = [\boldsymbol{m}_i^1\ \boldsymbol{m}_i^2 \ldots \boldsymbol{m}_i^l]$, where $\boldsymbol{m}_i^j$ and $\boldsymbol{s}^j$, $j = 1, 2, \ldots, l$, have the same dimensionality. Equation (1) can now be rewritten as

$$\phi_i(\boldsymbol{x}) = \prod_{j=1}^{l} \exp\left\{ -\frac{\left\| \boldsymbol{s}^j - \boldsymbol{m}_i^j \right\|^2}{\sigma_i^2} \right\}. \tag{3}$$

In (3), each basis function ϕ_i is represented as the product of the component Gaussian functions C_i^j, $j = 1, 2, \ldots, l$, where

$$C_i^j = \exp\left\{ -\frac{\left\| \boldsymbol{s}^j - \boldsymbol{m}_i^j \right\|^2}{\sigma_i^2} \right\}. \tag{4}$$

Thus, each C_i^j takes as an input the vector representing the features from a specific sensor j, so the output of C_i^j is related to the inputs obtained from the j^{th} sensor. Our objective is to eliminate the effect of the features generated from bad sensors. For the time being, let us *pretend* that we know the bad groups. Hence, we aim to design the component functions in such a manner that a component function C_i^k corresponding to a bad group will always take the value of unity (1) irrespective of the input $\boldsymbol{s}^k$. If we can do so, then C_i^k will *never* contribute anything to the total process, i.e., to $\phi_i(\boldsymbol{x})$. To achieve this, we design the component functions as

$$C_i^j = \left[\exp\left\{ -\frac{\left\| \boldsymbol{s}^j - \boldsymbol{m}_i^j \right\|^2}{\sigma_i^2} \right\} \right]^{1-e^{-\beta_j^2}}. \tag{5}$$

Clearly, in (5), if we set $|\beta_j| \approx 0$, then $C_i^j \approx 1$, thereby it can eliminate the effect of the sensor j in each of the basis function ϕ_i irrespective of values of $\boldsymbol{s}^j$. On the other hand, if $|\beta_j|$ is very large, then C_i^j in (5) reduces to C_i^j in (4) resulting in no change in the role of the basis functions. However, how do we know which group is good and which is bad? In other words, how do we set the values of the β_j's? The solution lies in the training process. We treat each β_j as an adjustable parameter and learn its appropriate value along with other parameters through training. With these preliminaries, we next discuss the network structure of GFSRBF.

A. Network Structure

GFSRBF network is a four-layer feedforward network as shown in Fig. 1. The network in Fig. 1 is designed for data obtained from three sensors, where two features are computed from each sensor. Also, it assumes three basis functions and two output nodes. The use of three basis functions has nothing to do with the number of sensors. We denote our training data set as $T = \{(\boldsymbol{x}, \boldsymbol{t}) | \boldsymbol{x} \in \Re^p, \boldsymbol{t} \in \Re^c\}$. Each point $\boldsymbol{x}$ has p features which can be divided into l groups and the total number of classes present is c. The division of the features into groups could be made based on sensors or by some other criteria. In our subsequent discussions, we denote the output of the ith layer by $z^{(i)}$. We now discuss the general structure of the network layer by layer.

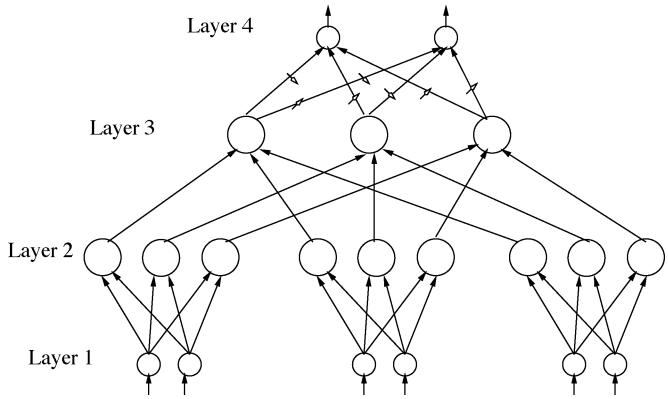

Fig. 1. GFSRBF network structure.

Layer 1) This layer is called the *input layer*. The number of nodes in this layer is equal to the dimensionality of the input data; here, it is p.

Layer 2) This layer is called the *component function layer*, and it is responsible for the feature selection task. If the network contains m basis functions, then this layer will have $l \times m$ nodes. Thus, this layer contains the component functions for each basis function for all feature groups. Let $z_{ij}^{(2)}$ denote the output of the component function related to the ith basis function and the jth group—the superscript "2" denotes the layer number. Then, we have

$$z_{ij}^{(2)} = \left[\exp\left\{ -\frac{\left\| \boldsymbol{s}^j - \boldsymbol{m}_i^j \right\|^2}{\sigma_i^2} \right\} \right]^{1-e^{-\beta_j^2}}. \tag{6}$$

In (6), β_j is an adjustable parameter related to the jth feature group. We call $\gamma_j = 1 - e^{-\beta_j{}^2}$, as the feature group modulator for the jth feature group. When $\gamma_j \to 0$, then $z_{ij}^{(2)} \to 1$. Thus, for a bad group of features *if* $\gamma_j \to 0$, then the effect of the jth group gets eliminated. The training procedure (to be discussed in Section II-B) will start with very low values of γ_j, i.e., with very small values of β_j^2 for all j, thereby making all feature groups unimportant. As the training process continues, the network allows features from only those groups which can lower the sum of square error significantly.

Layer 3) This layer is called the *basis function layer*; the number of nodes in this layer depends on the number of basis functions used (required) for solving the problem. The output of the ith basis function is

$$z_i^{(3)} = \prod_{j=1}^{l} z_{ij}^{(2)}. \tag{7}$$

Layer 4) This layer is called the *output layer*. The number of nodes in this layer is equal to the number of classes present in the data or the dimensionality of the output vector. The nodes in this layer are fully connected to the nodes of layer 3. The connection between node i in

layer 4 and node j in layer 3 bears a learnable weight w_{ij}. Like a conventional RBF network, the output of the ith node in this layer is given by

$$z_i^{(4)} = \sum_{j=1}^{m} w_{ij} z_j^{(3)} \tag{8}$$

where m is the number of nodes in layer 3. When the network is used for classification problems, then the target output of an output node lies in $[0,1]$, and this is true for all kinds of class labels that the data may have (probabilistic, possibilistic, fuzzy, or hard). However, (8) shows that $z_i^{(4)}$ is unbounded as the learnable weight w_{ij}, $\forall i,j$ can take any value. Consequently, for classification tasks, we modify the output of this node by adding a standard sigmoidal nonlinearity to this node function, so that the learning process becomes more simple. The output of node i in this layer is then computed as

$$z_i^{(4)} = \frac{1}{1+\exp\left(-\sum_{j=1}^{m} w_{ij} z_j^{(3)}\right)}. \tag{9}$$

For regression (function approximation) type of applications, nodes in the layer 4 use (8) while for classifier applications (9) is used. Now, we discuss the parameter updating strategies.

B. Learning Rules

We assume that there are c outputs and the training data contain points in $\Re^p$ along with its associated output in $\Re^c$. In case of classifiers, the output $\boldsymbol{t}$ is a label vector in $[0,1]^c$. Let the output associated with a data point $\boldsymbol{x}$ be $\boldsymbol{t} = [t_1\, t_2 \ldots t_c]^T$. Thus, we can define the instantaneous error for a data point $\boldsymbol{x}$ as

$$E_{\boldsymbol{x}} = \frac{1}{2}\sum_{i=1}^{c}\left(z_i^{(4)} - t_i\right)^2. \tag{10}$$

For our further discussion without loss of generality, we omit the subscript $\boldsymbol{x}$ and call the error term as E. The error function depends on the weights w_{ij}s connecting nodes of layers 2 and 3, the parameters of the basis functions, and the group feature modulators β_j's. We will consider fixed parameters for the basis functions, i.e., basis functions with fixed centers and spreads. We use the gradient–descent technique to update the weights and the feature group modulators β_j's. Thus, the update equations for w_{ij} and β_j are

$$w_{ij}(t+1) = w_{ij}(t) - \eta_w\left(\frac{\partial E}{\partial w_{ij}(t)}\right) \tag{11}$$

and

$$\beta_j(t+1) = \beta_j(t) - \eta_\beta\left(\frac{\partial E}{\partial \beta_j(t)}\right). \tag{12}$$

Here, η_w and η_β are predefined learning rates. For the *classification* network, i.e., for the network with sigmoidal transfer functions in the output units, we get

$$\frac{\partial E}{\partial w_{ij}} = z_j^{(3)} z_i^{(4)}\left(z_i^{(4)} - t_i\right)\left(1 - z_i^{(4)}\right) \tag{13}$$

and for the *regression* network, we get

$$\frac{\partial E}{\partial w_{ij}} = \left(z_i^{(4)} - t_i\right) z_j^{(3)}. \tag{14}$$

For both networks

$$\frac{\partial E}{\partial \beta_j} = -2\beta_j e^{-\beta_j^2}\sum_{k=1}^{m}\delta_k^{(3)} z_k^{(3)}\left(\frac{\left\|\boldsymbol{s}^j - \boldsymbol{m}_k^j\right\|^2}{\sigma_k^2}\right) \tag{15}$$

where

$$\delta_k^{(3)} = \frac{\partial E}{\partial z_k^{(3)}} = \sum_{i=1}^{c}\left(z_i^{(4)} - t_i\right) w_{ik}. \tag{16}$$

Note that along with w_{ij} and β_j the other parameters $\boldsymbol{m}_k^j$ and σ_k could also be learned using the gradient–descent technique. Since our objective here is to demonstrate the feature selection ability of the proposed scheme, we do not update $\boldsymbol{m}_k^j$ and σ_k, but we use judicious choices for them as discussed next.

C. Selection of Centers and Spreads

Selecting the parameters for the RBFs forms an important part in designing RBF networks. Generally, there are the following two common strategies used in practice.

1) The parameters for the basis functions are chosen *a priori* and are kept fixed. Only the weights between the hidden and output layers are updated during learning.
2) All parameters are optimized by a gradient–descent (or a similar) technique.

As discussed earlier, we follow the first strategy here. Initial centers and spreads are determined by running the fuzzy c means clustering algorithm (FCM) [4] on the training data. We use the fuzzifier $m = 2$ in all reported results. Once we obtain the cluster centers $\boldsymbol{\mu}_i$ by the FCM algorithm, we calculate the spread σ_i of the ith basis function as $\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$, where $\boldsymbol{\mu}_j$ is the center of the basis function nearest to $\boldsymbol{\mu}_i$. In other words

$$\sigma_i = \min_{j\neq i}\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|. \tag{17}$$

D. Threshold for the Feature Attenuators

The final values of the group feature attenuators (γ_j) of a trained GFSRBF network can be used to decide the importance of the sensors. A low value of γ_j indicates that sensor j is less important and a high value indicates a high importance of the sensor. In the limit, $\gamma_j = 0$ represents that the sensor j is very poor and it has derogatory effect on the system while $\gamma_j = 1$ suggests that the sensor j is very important. However, as γ_j's are modeled and updated, it can take any value in $[0,1]$. Here, we try to find a threshold th for γ_j, such that if γ_j takes values less than th we can discard sensor j.

From (6), we get

$$z_{ij}^{(2)} = \exp\left[-\gamma_j\frac{\left\|\boldsymbol{s}^j - \boldsymbol{m}_i^j\right\|^2}{\sigma_i^2}\right]. \tag{18}$$

We call a group of features bad, if all component basis functions (CBF) related to that feature group produce a response almost equal to one for all points. Specifically, we consider a feature group to be not important, if all CBFs produce a response greater than 0.95, even for points which are as far as 2σ distance away from the center of the associated CBF. Thus, we select that value of γ as the threshold which makes the right-hand side of (18) equal to 0.95 (≈ 1) when $\|\boldsymbol{s}^j - \boldsymbol{m}_i^j\|$ is replaced by 2σ. This way we obtain

$$e^{-4\gamma_j} = 0.95. \tag{19}$$

Equation (19) gives $\gamma_j = 0.0128$. Hence, we can safely discard a feature group with attenuation (γ) less than 0.01.

The GFSRBF network also has universal approximation properties, which is discussed in the Appendix.

III. GFSMLP Network

An MLP network can also be modified to do GFS. A feature selecting MLP was proposed by Pal and Chintalapudi [39] which used sigmoidal attenuation functions in the input layer. We generalize their idea to propose GFSMLP, a modified form of MLP for GFS.

Here, the philosophy is different from that used in case of RBF network. Let F_l be the attenuator function attached to the lth group of features. Each feature x_i of the lth group gets multiplied by the attenuator function F_l before it gets into the network. If $F_l = 0$, then no feature of the lth group will get into the network, while if $F_l = 1$, then every feature of the lth group enters the network unattenuated. For intermediate values of F_l, transformed values of the features enter into the network. Fig. 2 shows the architecture of a GFSMLP with just one hidden layer, and two groups of features with attenuation functions F_1 and F_2. Each attenuation function F_l should be such that it has a tunable parameter β_l and $F_l \in [0, 1]$. To facilitate the learning of β_l, F_l should be differentiable. Moreover, F_l should be such that over a reasonably large interval (a, b), as β_l goes from a to b, F_l should either monotonically increase from 0 to 1, or monotonically decrease from 1 to 0. There can be many choices for F_l; we take $F_l = e^{-\beta_l^2}$. β_l is a parameter related to the lth group of features. If $\beta_l \to 0$, then $F_l \to 1$ and if $\beta_l \to \pm\infty$, then $F_l \to 0$. The objective here is to select appropriate values of β_l's through training such that $F_l \to 1$, if l is associated with a useful group of features and $F_l \to 0$, if the lth group is a *bad* or *redundant* group. The parameters β_l can be learned by the backpropagation algorithm along with other parameters of an MLP.

Let us consider a network with sigmoidal activation functions and a single hidden layer. For each input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)^T \in \Re^p$, let S_l denote the set of features which belongs to the lth group. Let $z_i^{(1)}$, $z_i^{(2)}$, and $z_i^{(3)}$ be the outputs of the ith nodes of the input, hidden, and output layers, respectively. Thus, for input $\boldsymbol{x}$, if $x_i \in S_l$, the output of the ith node in the input layer would be

$$z_i^{(1)} = x_i F_l. \tag{20}$$

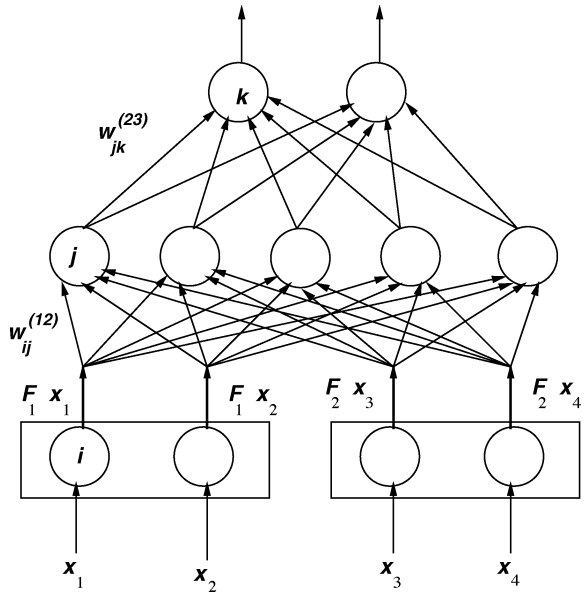

Fig. 2. MLP with GFS.

Let $w_{jk}^{(23)}$ be the weight of the link connecting the jth node of the hidden layer with the kth node of the output layer. Similarly, $w_{ij}^{(12)}$ denotes the weight connecting the ith node of the input layer with the jth node of the hidden layer. Thus

$$z_j^{(2)} = \frac{1}{1 + e^{-\sum_i z_i^{(1)} w_{ij}^{(12)}}} \tag{21}$$

and

$$z_k^{(3)} = \frac{1}{1 + e^{-\sum_j z_j^{(2)} w_{jk}^{(23)}}}. \tag{22}$$

For an input $\boldsymbol{x}$, if the target is $\boldsymbol{t}$, then we define the instantaneous error E as

$$E = \frac{1}{2} \sum_k \left(z_k^{(3)} - t_k \right)^2. \tag{23}$$

We define

$$\delta_i^{(q)} = \frac{\partial E}{\partial z_i^{(q)}}, \qquad q = 1, 2, 3. \tag{24}$$

Thus, the update equations for the two sets of weights can be derived as

$$w_{jk}^{(23)}(t+1) = w_{jk}^{(23)}(t) - \eta \delta_k^{(3)} z_j^{(2)} z_k^{(3)} \left(1 - z_k^{(3)}\right) \tag{25}$$

and

$$w_{ij}^{(12)}(t+1) = w_{ij}^{(12)}(t) - \eta z_j^{(2)} \left(1 - z_j^{(2)}\right) z_i^{(1)} \delta_j. \tag{26}$$

In (25) and (26), η is a predefined learning constant. The update equation for the feature attenuator of the lth group β_l can be derived as

$$\beta_l(t+1) = \beta_l(t) + \zeta \sum_{i \in S_l} 2\delta_i^{(1)} z_i^{(1)} \beta_l(t). \tag{27}$$

Here, ζ is also a predefined learning constant. The β_l's are so initialized that at the beginning of training no feature group is important (i.e., no feature gets into the network). As training continues, the β_l's of the groups which can reduce the error more will be changed significantly to make $F_l \rightarrow 1$.

Although we have shown the derivation for an MLP with only one hidden layer, its extension to MLPs with more than one hidden layer is straightforward. The GFSMLP also has the universal approximation property, which is proved in the Appendix.

IV. Experimental Results

We provide here experimental results on five data sets: Chem, Iris, RS-Data, Wine, and Breast Cancer.

The Chem data sat [48] is used to test the function approximation capability of the network. Chem contains data for operator's control of a chemical plant for producing a polymer by polymerization of some monomers. There are five input features, which a human operator may refer to for control and one output, that is his/her control. The input variables are monomer concentration (u_1), change of monomer concentration (u_2), monomer flow rate (u_3), and two local temperatures inside the plant (u_4 and u_5). The only output (y) is the set point for monomer flow rate. Chem contains a set of 70 data points obtained from an actual plant operation. In [48], it has been reported that the two local temperatures inside the plant, i.e., u_4 and u_5 *do not significantly contribute* to the output.

The remaining data are on classification problems. Iris data set [2] is a 4-D data set of 150 examples equally distributed in three classes. There are previous studies which suggest that two features among the four are enough for the classification task.

RS-Data [26] is generated from a 256 level satellite image of size 512×512 pixels captured by seven sensors operating in different spectral bands from Landsat-TM3. The 512×512 ground truth data provide the actual distribution of classes of objects captured in the image. This image is available along with full ground truth in the catalog of sample images of the ERDAS software and is used for testing various algorithms [26]. From this image, we produce the labeled data set where each pixel is represented by a 7-D feature vector and a class label. Each dimension of a feature vector comes from one channel. This data have eight classes representing different landcover types.

The Wine data set [6] consists of 178 data points in 13 dimensions distributed in three classes. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The Breast Cancer data set [6] consist of 699 points in 9-D distributed in two classes (malignant and benign).

In the following sections, we discuss in details the results obtained by GFSRBF and GFSMLP networks on these data sets. As stated earlier, the attenuation parameters for each network are initialized so that at the onset of training the network considers all feature groups to be unimportant. Thus, for GFSRBF, we set $\beta_j = 0.001$ which makes $\gamma_j \approx 0, \forall j$, where j represents a feature group. In GFSMLP, we set $\beta_j = 3$ thus making $F_j = 0.0001, \forall j$. For both GFSRBF and GFSMLP, we consider 0.01 as a threshold for the feature attenuators, i.e., if the value of γ_j corresponding to feature group j is less than 0.01, then we discard that feature group.

TABLE I
Values of γ_j in GFSRBF for Chem Data Set (Considering Three Groups of Features)

Group 1 $\{u_1, u_2\}$	Group 2 $\{u_3\}$	Group 3 $\{u_4, u_5\}$
0.00	1.00	0.00

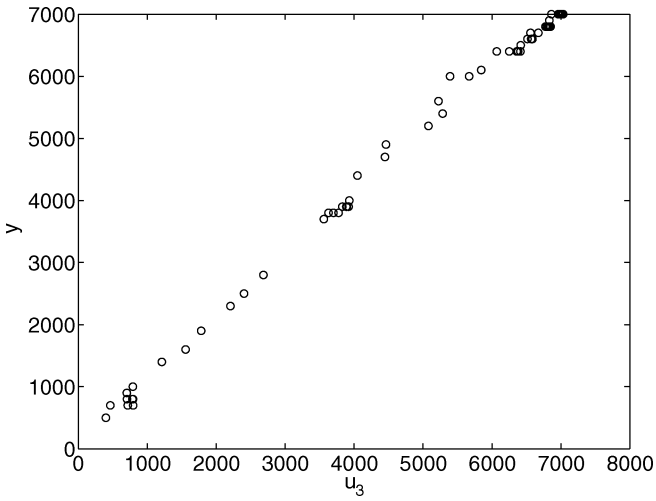

Fig. 3. Plot of y and u_3.

A. Chem

In this example, we demonstrate the feature selection capability of our network for the function approximation task. As stated earlier, the data set consists five input features. The five input features can be easily divided into three groups with respect to the type of information. The monomer concentration (u_1) and change of monomer concentration (u_2) can constitute one group, the monomer flow rate (u_3) can be a second group, and the temperature parameters (u_4 and u_5) can form the third group. With 15 basis functions, we find that the GFSRBF network accepts only the second group of features, i.e., only u_3 is important for the task (see Table I). Fig. 3 shows the plot of the output y with u_3. Fig. 3 reveals a very strong correlation between u_3 and y (correlation coefficient $= 0.9984$!). The Chem data set was used by Lin and Cunningham in [31]. To evaluate a system, they used a performance index (PI) defined as

$$\text{PI} = \frac{\sqrt{\sum_{k=1}^{n}(t_k - o_k)^2}}{\sum_{k=1}^{n}|t_k|}. \tag{28}$$

Here, t_k and o_k are the desired and actual outputs, respectively. They obtained a PI of 0.002245 on Chem by using features u_1, u_2, and u_3. The PI in our case was 0.004271.

A close look into the Chem data set shows that the values of feature 3 numerically dominate all other features. Table II shows the ranges of the input and output features. Since an RBF-type network computes the Euclidean norm, it is quite natural that features with larger numerical values dominate the output of the basis functions. Therefore, u_3 has the strongest influence on the network behavior. Moreover, u_3 has a strong positive correlation with the output y. Consequently, the network picks

TABLE II
RANGE OF FEATURE VALUES FOR CHEM DATA SET

Features	Minimum	Maximum
u_1	4.47	6.80
u_2	-0.29	0.17
u_3	401.00	7032.00
u_4	-0.40	0.20
u_5	-0.10	0.30
output	500.00	7000.00

TABLE III
PERFORMANCE OF GFSRBF ON NORMALIZED-CHEM

No. of basis functions	PI		No. of groups selected	
	Mean	Std. Dev.	Mean	Std. Dev.
2	0.0023	0.0000	2.0	0.00
3	0.0020	0.0000	1.8	0.44
5	0.0037	0.0001	1.0	0.00
7	0.0029	0.0002	2.0	0.00
10	0.0025	0.0005	2.0	0.00
15	0.0020	0.0000	1.0	0.00

up u_3. However, previously, it has been reported that the features u_1 and u_2 also have some effect on the output [48]. Our network cannot detect that, and as a result we get reasonable (but not very good) performance as suggested by the PI value. This is not a problem of the model or of the philosophy being used, but is due to very wide variance of different features. To establish this fact, we normalize feature 3 (u_3) and the output (y) so that each of these two lie in $[0, 1]$. We call this new data set as normalized-Chem.

With normalized-Chem, we run GFSRBF with different number of basis functions and different initializations of the FCM algorithm. The FCM outputs are used to compute the centers and spreads of the basis functions. For a fixed number of basis functions, various FCM initializations do not significantly change the feature attenuators and the performance. Table III gives the average performance of GFSRBF on normalized-Chem data set for different number of basis functions. With a fixed architecture (a fixed number of basis functions), five independent runs are made with different initializations, and the average value of PI and the number of groups selected are shown in Table III. The frequency of each of the groups selected in these 30 runs are shown in Fig. 4. From Fig. 4, we see that the network rejects the third group for most of the runs. From Table III, we see that the best performance is obtained by using three basis functions and 15 basis functions. In case of three basis functions, the network considers the first and the second group of features, but for 15 basis functions, it considers only the second group. In both cases, the networks result in almost the same average performance. This establishes that changing the number of basis functions changes the learning machine, so the importance of the features may also vary.

The performance of GFSMLP on normalized-Chem data set for different hidden nodes is shown in Table IV. Here too, the average performance in terms of PI and the number of groups selected are shown for five independent runs for each architecture. Table IV shows that the GFSMLP selects two groups for all runs. The two groups selected are the first and the second group. The PI value suggests that GFSMLP can also do the function approximation job with a good accuracy. Note that GFSMLP gives relatively more importance on the first group than GFSRBF. This emphasizes the fact that importance of a feature (or a group of features) is a function of the tool being used to solve a problem.

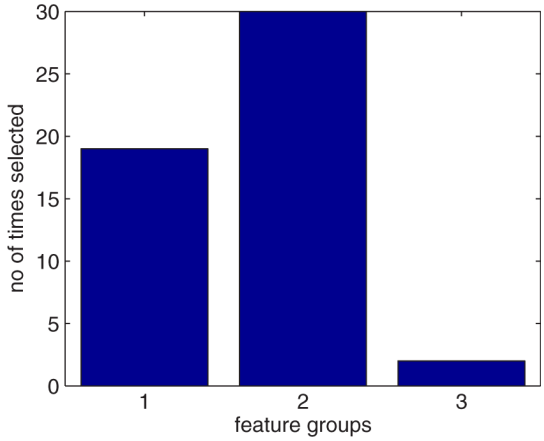

Fig. 4. Number of times each feature was selected for Chem data set using GFSRBF.

TABLE IV
VALUES OF F_j IN GFSMLP FOR NORMALIZED-CHEM DATA SET (CONSIDERING THREE GROUPS OF FEATURES)

No. of hidden nodes	PI		No. of groups selected	
	Mean	Std. Dev.	Mean	Std. Dev.
5	0.002360	0.0000	2.0	0.0
7	0.002295	0.0000	2.0	0.0
10	0.002258	0.0000	2.0	0.0
15	0.002022	0.0000	2.0	0.0

TABLE V
VALUES OF γ_j FOR IRIS DATA SET (CONSIDERING FOUR GROUPS OF FEATURES) USING GFSRBF

Group 1 (f_1)	Group 2 (f_2)	Group 3 (f_3)	Group 4 (f_4)
0.00	0.00	0.33	0.99

B. Results on Iris Data Set

For this data set, we select 100 points randomly as the training data. First, we assume that the four features of Iris form four groups with one feature in each group, so we obtain a feature modulator for each feature. We use a network with six basis functions. The values of the modulator functions for all features are shown in Table V, which clearly shows that the network does not accept the first and the second features. This result is consistent with the well-known fact that the third and fourth features of Iris data set are enough for the classification task. The number of misclassifications obtained on the training data is three and on the whole data (150 points) is five. This performance is quite comparable with that of other classifiers [4].

Physically, the Iris features are the sepal length (f_1), sepal width (f_2), petal length (f_3), and petal width (f_4) of Iris flower. Therefore, we can make two natural groups of features, i.e., one group characterizing the sepals and the other containing the petals. In other words, we consider features 1 and 2 as the first group and features 3 and 4 as the second group. With this grouping, we ran GFSRBF for different basis functions and dif-

TABLE VI
MISCLASSIFICATIONS AND NUMBER OF GROUPS SELECTED FOR IRIS DATA SET WITH GFSRBF (CONSIDERING TWO GROUPS OF FEATURES)

No. of basis functions	Trng Error (%)		Test Error (%)		No. of groups selected	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
3	3.0	0.0	4.0	0.0	1.0	0.0
5	3.0	0.0	4.0	0.0	1.0	0.0
7	3.0	0.0	4.0	0.0	1.0	0.0
10	3.0	0.0	2.8	1.08	1.0	0.0

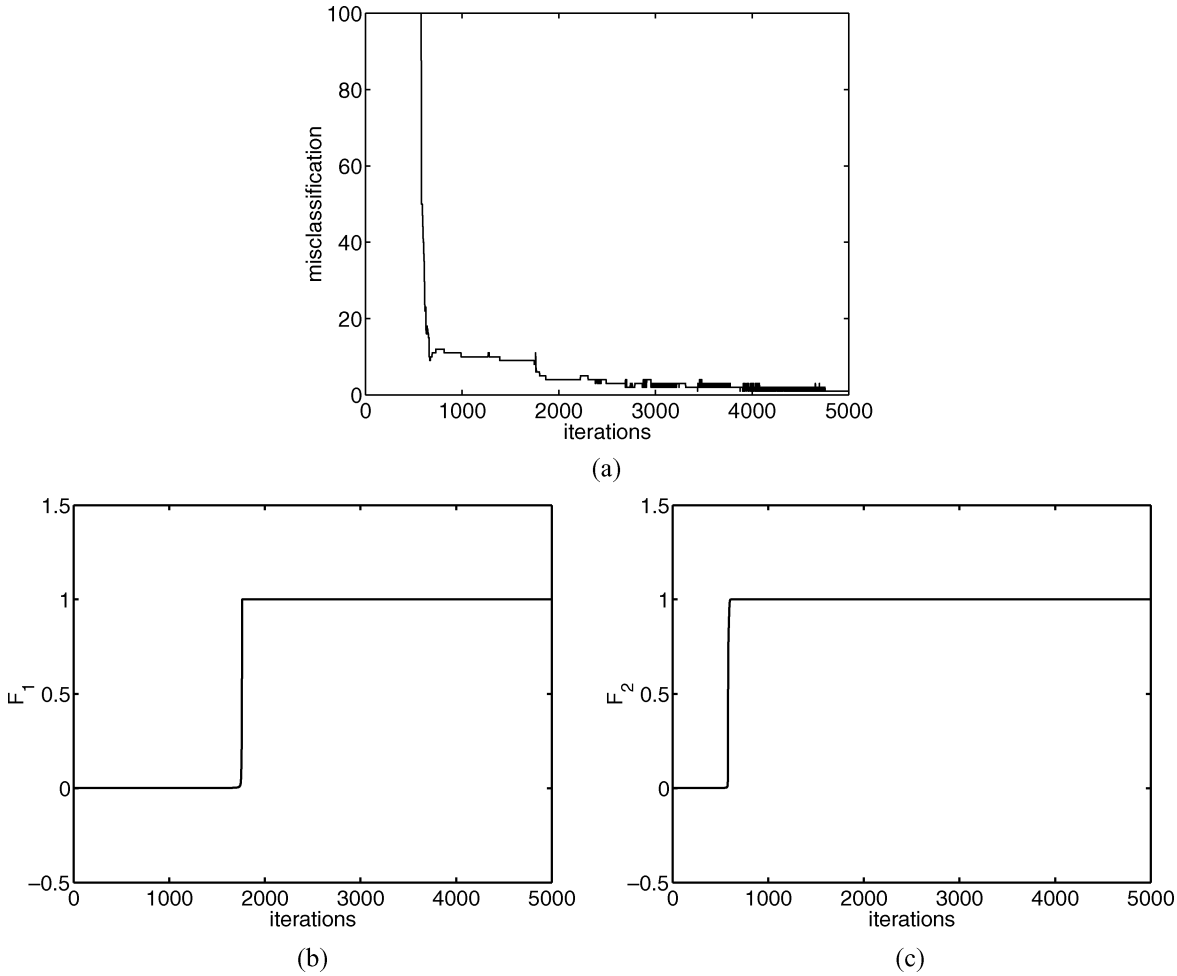

Fig. 5. Variation of attenuator values and misclassification with number of iterations for Iris data set: (a) misclassification, (b) attenuator values for group 1 features, and (c) attenuator values for group 2 features.

ferent FCM initializations. Table VI shows the average misclassifications and the average number of groups selected for five independent runs of GFSRBF for each architecture. Table VI clearly suggests that our network selects only one group irrespective of the number of basis functions used. Also, in all cases, the network selected group 2 (i.e., f_3 and f_4) features.

GFSMLP can also select the relevant features for the Iris data set. We use a GFSMLP with ten nodes in the hidden layer, each with a sigmoidal activation function. We use the same 100 samples for training as used for GFSRBF. In Fig. 5(a)–(c), we show the performance of our feature selection algorithm in a pictorial manner. Fig. 5(a) gives the variation of the misclassifications with the number of iterations for a typical run. Fig. 5(b) and (c) depicts the feature attenuator values (F_l) for different iterations for groups 1 and 2, respectively, for the same run. Fig. 5(a) shows that the misclassification drops sharply from 100 to 10 at around 800 iterations. Fig. 5(c) reveals that at that time the features in the second group enter the network. As the training continues, we find that again there is a sharp decrease in misclassification around 1800 iterations when the first group of features gets in the network [Fig. 5(b)]. This behavior is consistent with most of the runs. We find that the second group of features first gets in the network to give an average misclassification of 10 (averaged over ten runs of the same network with different initializations). If training is continued, then the first group of features also gets in and reduces the misclassifications to 0. The final network produces a misclassification of 1 on the whole data. This clearly demonstrates the feature selection capability of the network. It suggests that features 3 and 4 constitute a very important group of features for Iris, but the other group also

TABLE VII
VALUES OF γ_j IN GFSRBF FOR RS-DATA (CONSIDERING SEVEN GROUPS, ONE FEATURE PER GROUP)

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
0.42	0.00	0.84	0.99	0.41	0.99	0.25

TABLE VIII
MISCLASSIFICATIONS AND NUMBER OF GROUPS OF FEATURES SELECTED FOR RS14 DATA SET WITH GFSRBF

No. of basis functions	Trng Error %		Test Error %		No. of groups selected	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
10	24.72	0.23	17.55	0.28	5.6	0.89
20	20.61	0.71	16.42	0.34	6.0	0.00
25	19.93	1.11	14.96	0.10	5.4	0.54
30	18.96	0.73	15.21	0.75	5.2	0.44

has some discriminating ability that can facilitate the learning in MLP. In fact, this observation is consistent with the results reported elsewhere [39] which suggests that features 1 and 3 have equally good discriminating power as features 3 and 4. Here, we like to emphasize the fact that this network is primarily intended to select features. Once the features are selected, one can remove the nodes and links associated with the bad or redundant features and retrain the net for a few more epochs for further improving the performance of the network. For example, in this particular case, after features 3 and 4 are selected, one can delete the links associated with features 1 and 2 and retrain the net to achieve a better performance.

C. RS-Data

This data set contains 262 144 points distributed in eight classes. In previous studies with this data set [26], [28], training-test partitions were created in the following manner. From each class, 200 points were randomly selected to get a training sample of 1600 points, and the rest of the data was used for testing. In our study, we also use the same protocols to generate the training-test partition. For this data set, we initially consider each of the seven features as constituting a group. Running GFSRBF with 30 basis functions, we obtain a misclassification of 18.43% on the training data and 15.59% on the test data. The final values of the feature attenuators are shown in Table VII, which reveals that the network completely discards the second feature. In [26], a misclassification of 21.8% was obtained on the test data. In [28], a misclassification of about 14% on the test data was reported. The performance of our system is comparable to those, though our system uses smaller number of features.

In another experiment, we generated an additional feature from each of the seven channels of the image. For each pixel p in the image, we considered its eight-neighborhood over a 3×3 window and computed the standard deviation of the 9-pixel values (the neighborhood of p and the pixel itself); we call it d_p. In the new data set, for each channel, we take the gray value of p and d_p as features, so we have 14 features divided into seven groups. We call this the RS14 data set. Table VIII shows the average misclassification (in percent) on training and test data for five independent runs for different architectures. Fig. 6(a) shows the number of times each feature gets selected for these 20 runs. From Fig. 6(a), we find that the features from the second sensor are selected the least number of times. This is consistent with the results described in Table VII using seven features.

No previous study regarding the goodness of features of RS-Data exists. We made a naive feature analysis to compare our results. We ran the k-nearest neighbor classifier [4] on this data with all possible combination of six features, i.e., in each run, we left out one feature. Among the seven possible combinations, the feature set $\{1, 3, 4, 5, 6, 7\}$ results in the least number of misclassifications. This clearly points out that feature 2 is a poor feature.

The GFSMLP selects smaller number of groups from RS14 data set than GFSRBF. Table IX shows the percentage of misclassification and the number of groups of features selected for different number of hidden nodes for GFSMLP. For each architecture (a fixed number of hidden nodes), the average misclassifications and average number of features selected for five independent runs are reported in Table IX, from where it is evident that GFSMLP produces a poorer classification than GFSRBF. However, for all cases, our results are better than the result reported in [26]. In [26], a misclassification of 21.8% is reported on the test data. Fig. 6(b) shows the frequencies with which various groups are selected by GFSMLP. Here too, the features from the second sensor are selected the least number of times. Hence, this result is also consistent with the results of the previous experiments on RS-Data.

D. Wine

For Wine data set, a natural grouping of features was not possible. Thus, we considered 13 groups with one feature in each group. We used 100 randomly chosen points from the data set for training and the remaining 78 points for testing.

We trained GFSRBFs with different number of basis functions. As shown in the first column of Table X, we considered five different architectures of GFSRBF. For each architecture, we made ten independent runs of the network with different initializations, keeping the training-test partition fixed. Table X shows the average misclassifications along with their standard deviations for ten independent runs of GFSRBFs with each architecture. Similarly, for GFSMLP, we also considered five different architectures as shown in the first column of Table XI. For each architecture, we made ten independent runs. The summary of the runs by GFSMLP is included in Table XI. Tables X and XI show that the number of features selected by GFSMLP is always lower than that selected by GFSRBF. Fig. 7(a) and (b) shows the

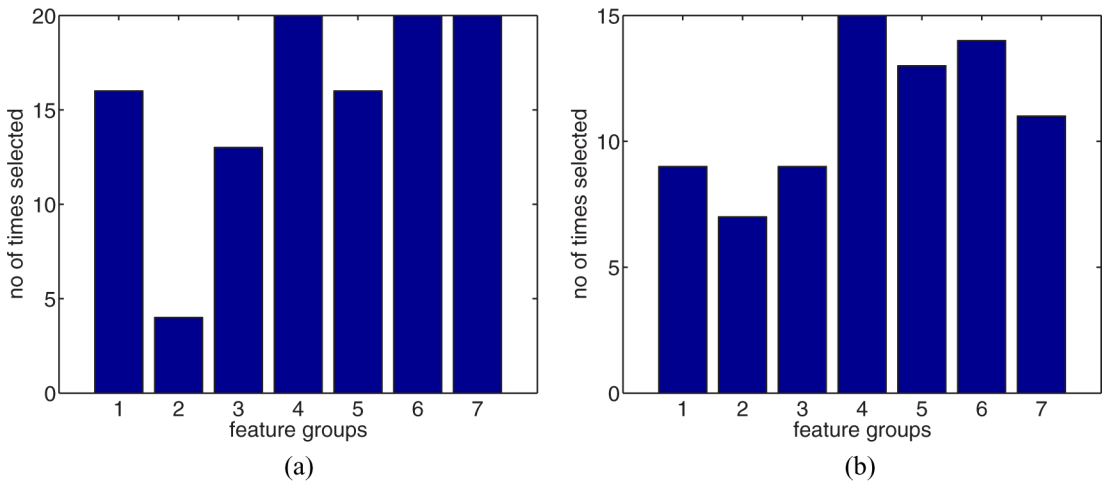

Fig. 6. Bar diagram showing the number of times each feature was selected for RS14 data set: (a) GFSRBF and (b) GFSMLP.

TABLE IX
MISCLASSIFICATIONS AND NUMBER OF GROUPS OF FEATURES SELECTED FOR RS14 DATA SET WITH GFSMLP

No. of hidden nodes	Trng Error %		Test Error %		No. of groups selected	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
10	26.52	4.25	20.88	2.53	3.4	0.54
20	21.82	2.67	20.76	3.12	3.8	0.44
25	18.02	2.26	17.96	1.38	4.2	0.44
30	17.63	2.74	18.86	1.65	4.0	0.70

TABLE X
MISCLASSIFICATIONS AND NUMBER OF FEATURES SELECTED FOR WINE DATA SET WITH GFSRBF

No of basis functions	Trng Error %		Test Error %		No. of features selected	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
3	4.5000	0.5270	5.1280	0	7.5000	0.5270
5	7.0000	0	4.8716	1.3240	6.0000	0
7	3.0000	0	1.2820	0	5.4000	0.8433
10	4.0000	0	2.5640	0	6.0000	1.0541
15	4.0000	0	2.5640	0	5.9000	0.3162

TABLE XI
MISCLASSIFICATIONS AND NUMBER OF FEATURES SELECTED FOR WINE DATA SET WITH GFSMLP

No.of hidden nodes	Trng Error %		Test Error %		Features Selected	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
3	4.3000	1.3375	8.4612	1.5048	5.8000	0.6325
5	1.9000	1.1972	4.6153	1.9301	4.9000	0.5676
7	1.9000	1.1972	4.4871	2.2814	5.2000	1.2293
10	4.4000	1.9551	3.5897	3.1287	4.1000	0.7379
15	2.4000	1.4298	7.1794	2.7826	4.1000	0.8756

frequency distribution of the number of times each feature is selected over 50 runs of GFSRBF and GFSMLP, respectively.

In [29], the average misclassification obtained after feature selection using three different feature selection methods is reported. They report results with three methods called relief (a feature-weight-based statistical approach), IFN (an information theoretic feature selection scheme), and ABB (a breadth first search, backward selection algorithm). With Wine data set, they report an average test error of 8.3%, 5.0%, and 21.7% on the reduced set of features using IFN, relief, and ABB, respectively. They use a decision tree as the classifier. With GFSRBF, we obtain a mean test error (the mean of the entries of the fourth column of Table X) of 3.28%, and with GFSMLP, we obtain a mean test error (the mean of the entries of the fourth column of Table XI) of 5.66%. This shows that our method produces comparable results with other state-of-the-art classifiers and feature selection methods.

E. Breast Cancer

For the Breast Cancer data set, we randomly selected 500 points from the 699 points to use them as the training set, and the rest are used for testing. For this data set, we also tested the performance of GFSRBF using different architectures. We considered networks with 5, 7, 10, and 15 basis functions. Table XII

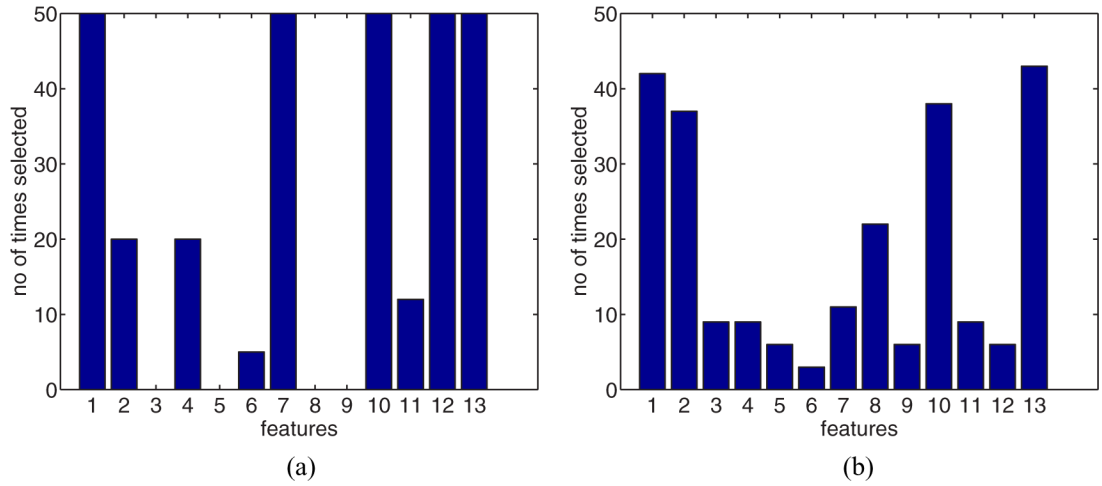

Fig. 7. Bar diagram showing the number of times each feature was selected for Wine data set: (a) GFSRBF and (b) GFSMLP.

TABLE XII
MISCLASSIFICATIONS AND NUMBER OF FEATURES SELECTED FOR BREAST CANCER DATA SET WITH GFSRBF

No. of basis functions	Trng Error %		Test Error %		Features Selected	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
5	4.06	0.3134	1.004	0.75	6.3000	0.6749
7	4.14	0.2319	1.14	0.24	6.4000	0.5164
10	3.90	0.2867	1.14	0.24	6.7000	0.8233
15	3.96	0.2065	1.00	0.00	6.1000	0.3162

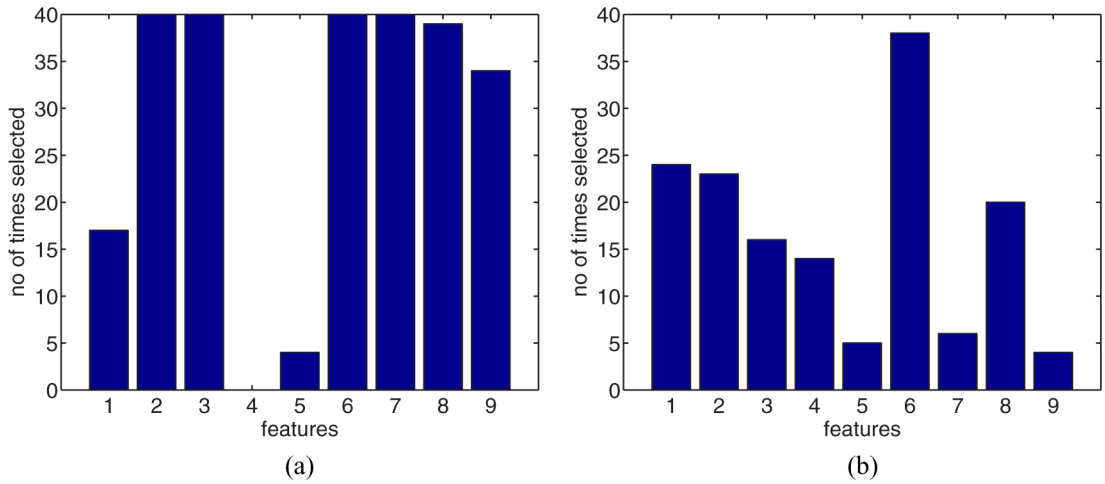

Fig. 8 Bar diagram showing the number of times each feature was selected for Breast Cancer data set: (a) GFSRBF and (b) GFSMLP.

shows the mean misclassification and standard deviation for ten independent runs of GFSRBF for each architecture. It also shows the average number of features selected.

Table XII reveals that the performance of GFSRBF on this data in terms of misclassification and the number of features selected does not vary much with the change in the number of basis functions. The bar diagram in Fig. 8(a) shows the number of times each feature is selected for 40 runs.

Table XIII shows the performance of GFSMLP on the Breast Cancer data set. Here also, the same protocol is followed as in case of the GFSRBF network. We considered GFSMLPs with 5, 7, 10, and 15 hidden nodes. Here too, we find that the performance is quite stable with respect to changes in the number of hidden nodes. Fig. 8(b) depicts the frequency of the selected features over 40 runs. Fig. 8 reveals that GFSRBF completely rejects feature 4, while features 2,3,6, and 7 are always selected. On the other hand, for GFSMLP, feature 9 is the least important, and although feature 6 appears to be the most important, there is no feature which is always picked up by the network. This reemphasizes that utility of a feature is dependent on the machine learning tool that is used to solve the problem.

Here too, we compare our methods with the results reported in [29]. In [29], a test error of 6.0%, 6.4%, and 6.4% is reported on the Breast Cancer data set using a reduced set of features obtained by IFN, relief, and ABB, respectively. Using GFSRBF, we obtain a mean misclassification (mean of the entries in the

TABLE XIII
MISCLASSIFICATIONS AND NUMBER OF FEATURES SELECTED FOR BREAST CANCER DATA SET WITH GFSMLP

No. of hidden nodes	Trng Error %		Test Error %		Features Selected	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
5	4.32	0.54	1.15	0.33	3.5000	0.5270
7	4.16	0.40	1.80	0.91	3.9000	0.8756
10	3.56	0.47	1.25	0.54	4.1000	0.7379
15	3.86	0.48	0.95	0.59	3.6000	0.8433

TABLE XIV
MISCLASSIFICATIONS PRODUCED BY AN ORDINARY MLP WITH VARIOUS SETS OF FEATURES ON BREAST CANCER DATA SET

No. of hidden nodes	All Features		GFSMLP selected		GFSRBF selected	
	Trng. Error	Test Error %	Trng Error %	Test Error %	Trng. Error %	Test Error %
5	1.00 ± 1.44	2.80 ± 1.78	1.40 ± 0.51	1.2 ± 0.83	1.00 ± 0.24	2.3 ± 0.97
7	0.76 ± 0.86	1.90 ± 0.74	0.84 ± 0.33	1.6 ± 0.82	0.84 ± 0.45	2.3 ± 0.70
10	0.36 ± 0.54	2.30 ± 1.03	2.4 ± 0.56	2.1 ± 0.65	0.12 ± 0.19	3.3 ± 0.27
15	0.20 ± 0.24	2.30 ± 1.25	2.2 ± 0.35	1.9 ± 1.24	0.20 ± 0.34	2.4 ± 1.29
Mean		2.32		1.7		2.57

TABLE XV
MISCLASSIFICATIONS PRODUCED BY AN ORDINARY RBF NETWORK WITH VARIOUS SETS OF FEATURES ON BREAST CANCER DATA SET

No. of basis functions	All Features			GFSMLP selected			GFSRBF selected		
	Trng. Error (%)	Test Error (%)	spread	Trng Error (%)	Test Error (%)	spread	Trng. Error (%)	Test Error (%)	spread
5	3.80	1.50	6.4	4.00	1.50	2.7	3.60	1.00	6.2
7	3.80	1.00	6.4	3.80	1.00	4.9	3.60	1.00	6.0
10	3.60	0.00	8.8	3.60	1.00	4.8	3.40	1.00	5.7
15	3.40	0.50	6.7	3.40	0.00	2.7	3.60	0.50	5.6
Mean		0.75			0.87			0.87	

fourth column of Table XII) of 1.071%, and using GFSRBF, we obtain a mean misclassification (mean of the entries in the fourth column of Table XII) of 1.28%.

F. Evaluation of Features

We now try to evaluate the quality of features that are selected by GFSRBF and GFSMLP. We use the Wine and Breast Cancer data set for this purpose. We again demonstrate here that the suitability of the features depends not only on the data but also on the learning machine. We train conventional MLPs and conventional RBFs with all the features present in the data and also with the features selected by GFSMLP and GFSRBF and compare their performance.

For the Wine data set, a GFSRBF selects, on average, 6.16 features. Thus, we take the six most frequently selected features as shown in Fig. 7(a) as the features selected by GFSRBF for the Wine data set. These selected features are 1, 2, 7, 10, 12, and 13. For the Wine data set, the GFSMLP selects 4.82 (≈ 5) features on average. Thus, the GFSMLP selected features are 1, 2, 8, 10, and 13 [as evident from Fig. 7(b)]. Similarly, we get the GFSRBF selected features for the Breast Cancer data set as 1, 2, 3, 6, 7, 8, and 9 and the GFSMLP selected features as 1, 2, 6, and 8.

For the experiments, we use the same training test partitions as used in the previous experiments. We use the Mathworks neural network toolbox implementations of RBF and MLP. For MLP, we use sigmoidal activation functions and the *trainlm* algorithm for training. The Mathworks neural network toolbox implementation of RBF takes the same spread for all basis functions. We experiment with spreads ranging from 0.1 to 10 with an increment of 0.1 and report the best result that we obtain.

Table XIV depicts the results of a conventional MLP when run with different hidden nodes on different sets of features of the Breast Cancer data set. For each architecture, five independent runs are made, and the average percentage of misclassifications on the training and test data along with the standard deviation are reported in Table XIV. The last row of Table XIV shows the mean test errors using the different sets of features. Table XIV clearly shows that an ordinary MLP produces smaller test error on the GFSMLP features for all four different architectures that we have tried. When MLPs are trained with features selected by GFSRBF, the test result is slightly worse than what is achieved with all features. This reemphasizes the fact that features selected by GFSRBF are good, but they are the best with RBF. Table XV gives the results of a conventional RBF on the same set of features. Here, we see that the performance of an RBF network is almost the same on both features selected by GFSRBF and GFSMLP.

Tables XVI and XVII display the results on Wine data set for MLP and RBF on different sets of features. As expected, here too, we notice that, on average, an ordinary MLP performs better with the GFSMLP selected features and an ordinary RBF network performs better with the features selected by GFSRBF.

V. DISCUSSIONS

We discuss some of the features and limitations of the methods proposed here. In order to analyze the behavior of

TABLE XVI
MISCLASSIFICATIONS PRODUCED BY AN ORDINARY MLP WITH VARIOUS SETS OF FEATURES ON WINE DATA SET

No. of hidden nodes	All Features		GFSMLP selected		GFSRBF selected	
	Trng. Error %	Test Error %	Trng Error %	Test Error %	Trng. Error %	Test Error %
5	0.0 ± 0.0	1.28 ± 1.39	1.2 ± 0.83	3.07 ± 1.14	0.4 ± 0.54	4.35 ± 1.46
7	0.0 ± 0.0	1.53 ± 1.28	0.2 ± 0.44	4.10 ± 1.06	0.0 ± 0.0	4.35 ± 2.65
10	0.0 ± 0.0	2.05 ± 3.21	0.2 ± 0.44	3.84 ± 0.89	0.0 ± 0.0	5.64 ± 1.96
15	0.0 ± 0.0	2.56 ± 2.39	0.0 ± 0.00	4.61 ± 1.46	0.0 ± 0.0	3.33 ± 1.46
Mean		1.85		3.90		4.42

TABLE XVII
MISCLASSIFICATIONS PRODUCED BY AN ORDINARY RBF NETWORK WITH VARIOUS SETS OF FEATURES ON WINE DATA SET

No. of basis functions	All Features			GFSMLP selected			GFSRBF selected		
	Trng. Error (%)	Test Error (%)	spread	Trng Error(%)	Test Error(%)	spread	Trng. Error(%)	Test Error(%)	spread
5	1.00	3.84	1.4	5	3.84	0.5	2.00	1.28	0.5
7	1.00	2.56	1.1	5	3.84	0.5	2.00	3.84	0.4
10	1.00	3.84	0.8	4	2.56	0.4	2.00	3.84	0.3
15	0.00	1.28	0.9	2	7.69	0.4	1.00	2.56	0.4
Mean		2.88			4.48			2.88	

our schemes, as discussed in the Introduction, we characterize features (or groups of features) as essential features, redundant features, derogatory or bad features, and indifferent features. Essential features are those that are necessary for the task at hand and any reasonable feature selection method should select them. The bad features are those which hinder the learning and must be discarded. By redundant features, we refer to those features, all of which may not be needed for the task at hand, but some of them are required. In other words, we can consider the set of redundant features as a set of good features, but not all of them are required to solve the problem. The indifferent features are those which neither help nor cause any problem in learning. For example, if a feature has almost constant value for all samples in the training set, then it is an indifferent feature. Next, we provide an example to illustrate our classification of the features.

Example: Let us define a four-class problem with five features: sex, height, weight, eye color, and number of legs. The four classes are characterized as follows:

Class 1) male and short height or low weight;
Class 2) male and long height or heavy weight;
Class 3) female and short height or low weight;
Class 4) female and long height or heavy weight.

From the description of the four classes, it is easy to see that *sex* is an essential feature and *height* and *weight* are *good* features, but if one of them is selected then the other is redundant. The *eye color* is a bad feature as it does have a relation to the class definitions and it does not help to identify the classes but adds the dimension of the problem. The *number of legs* is an indifferent feature, as it takes the same value for all individuals/data points.

Hence, the essential features and some of the good ones can solve the problem and there could be more than one such choice.

With these definitions in mind we review the behavior of our networks. The proposed methods are primarily dependent on gradient search. If a group of features can reduce the error faster, its associated modulator is expected to change faster to enhance its influence on the training error. Thus, if the training error is low, we can state the following.

- Essential features are selected by the network; otherwise, the error cannot be low.
- Derogatory/bad features (the features which hinder the learning process) are not selected by the network because the error cannot be low. Bad features cannot reduce the training error; rather, they may increase the error. If the training tries to set the modulators associated with bad features to high values, then the error will start increasing unless all weights associated with those features are set to practically zero values.
- Since indifferent features cannot reduce the error, the modulator values are not expected to change. Thus, these features will not get into the network because when the training starts all features are treated as not important.
- Some redundant/correlated features, however, may get into the network as we do not penalize the network if it selects more features.

To summarize, we can say that the proposed schemes will be quite effective in selecting essential features and eliminating bad and indifferent features, but the selected feature set may contain some correlated (redundant) features. Moreover, depending on the initialization, the training may converge to a poor minimum (very high training error) and in that case the selected features are not likely to be good ones. Usually, such a situation does not arise but if it does, it is not difficult to detect and discard that solution. Finally, if there are several subsets of features or several subsets of sensors that can solve the problem equally well, our system can pick up any one of them depending on the initialization. In other words, depending on the initialization, different set of features/sensors may be selected by the system in different runs. This phenomenon is evident from some of our experiments, for example, from Figs. 4 and 6–8, we see that the network selects different sets of features in different runs but each run produces an acceptable and comparable training error.

VI. CONCLUSION

Many real life applications use data from several sensors for decision making. Intelligent systems for automatic inspection and controlling of welding, medical diagnosis, and controlling

of range safety for missile testing are some such examples. Typically, each sensor output is converted into a set of features. For example, X-ray radiograph may be used to compute a set of features for weld inspection. More sensors mean more cost, more processing time, and sometimes more hazard (X-rays), and they do not necessarily lead to a better performance. Therefore, if we can reduce the number of required sensors, we can save cost, time, and design complexity, and sometimes minimize the risk. This is a very important problem but not addressed in literature. In this paper, we provide some novel solutions to this problem. In particular, we achieved the following.

1) We proposed two schemes for solving these problems. One scheme is based on the RBF framework and the other uses MLP.
2) Both schemes are capable of selecting useful groups of features (sensors).
3) Both schemes can also select individual features.
4) We proved that both GFSRBF and GFSMLP have the universal approximation property.
5) Our experimental results reconfirm that the importance of features depends on the tool used to solve a problem.
6) Our limited experiments show that GFSMLP usually needs lower number of features to do a task than GFSRBF.

In near future, we would like to extend this concept to the neurofuzzy framework.

APPENDIX
UNIVERSAL APPROXIMATION PROPERTY OF GFSRBF

The universal approximation property of RBF is well known [16], [40]. If $\gamma_j = 1$, then GFSRBF reduces to RBF. However, during the training process, γ_j usually takes values in $[0, 1]$. Therefore, it is necessary to check the universal approximation property of GFSRBF. Unless, GFSRBF has the universal approximation property, it may not be able to learn the input–output relation for all functions and thus may not be able to do the GFS task, so we check this property here.

We consider the GFSRBF network for function approximation (i.e., the one without the sigmoidal nonlinearity in the output node) with a single output. The proof can be easily extended for the multiple output case.

Definition: Let $X \subset \Re^p$ and G be a family of functions on X with values in $\Re$. Suppose that for all $\boldsymbol{x}, \boldsymbol{y} \in X$, such that $\boldsymbol{x} \neq \boldsymbol{y}$, there is a function $f \in G$ such that $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$; then we say that G is a separating family of functions on X [17].

Let $\boldsymbol{x} = (\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^l)$. We define a function family Φ as

$$\Phi = \left\{ \prod_{j=1}^{l} \left[\exp\left\{ -\frac{\|\boldsymbol{s}^j - \boldsymbol{m}^j\|^2}{\sigma^2} \right\} \right]^{\gamma_j} \middle| \boldsymbol{s}^j, \boldsymbol{m}^j \in \Re^{p_j}, \sum_{j=1}^{l} p_j = p, \gamma_j \in [0,1], \sigma \in \Re \right\}.$$

Next, we prove a few lemmas concerning the function family Φ.

Lemma 1: Φ is a separating family.

Proof: For any $\boldsymbol{x}_1 = (\boldsymbol{s}_1^1, \boldsymbol{s}_1^2, \ldots, \boldsymbol{s}_1^l)$ and $\boldsymbol{x}_2 = (\boldsymbol{s}_2^1, \boldsymbol{s}_2^2, \ldots, \boldsymbol{s}_2^l) \in X$, if $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, then there exists an i, $0 < i \leq l$, such that $\boldsymbol{s}_1^i \neq \boldsymbol{s}_2^i$. Without loss of generality, we assume $\boldsymbol{s}_1^1 \neq \boldsymbol{s}_2^1$. Pick any $\phi \in \Phi$ with $\gamma_j \neq 0$. If $\phi(\boldsymbol{x}_1) \neq \phi(\boldsymbol{x}_2)$, then it is done. If $\phi(\boldsymbol{x}_1) = \phi(\boldsymbol{x}_2)$, let

$$\phi(\boldsymbol{x}_1) = \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_1^1 - \boldsymbol{m}^1\right\|^2}{\sigma^2} \right\} \right]^{\gamma_1} \times \prod_{j=2}^{l} \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_1^j - \boldsymbol{m}^j\right\|^2}{\sigma^2} \right\} \right]^{\gamma_j}$$

and

$$\phi(\boldsymbol{x}_2) = \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_2^1 - \boldsymbol{m}^1\right\|^2}{\sigma^2} \right\} \right]^{\gamma_1} \times \prod_{j=2}^{l} \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_2^j - \boldsymbol{m}^j\right\|^2}{\sigma^2} \right\} \right]^{\gamma_j}.$$

If

$$\left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_1^1 - \boldsymbol{m}^1\right\|^2}{\sigma^2} \right\} \right]^{\gamma_1} = \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_2^1 - \boldsymbol{m}^1\right\|^2}{\sigma^2} \right\} \right]^{\gamma_1}$$

then $\boldsymbol{m}^1$ is equidistant from both $\boldsymbol{s}_1^1$ and $\boldsymbol{s}_2^1$. Let $\hat{\boldsymbol{m}}^1$ be such that it is not equidistant from both $\boldsymbol{s}_1^1$ and $\boldsymbol{s}_2^1$. As $\boldsymbol{s}_1^1 \neq \boldsymbol{s}_2^1$, then $\hat{\boldsymbol{m}}^1$ always exists. In ϕ, replace $\boldsymbol{m}^1$ by $\hat{\boldsymbol{m}}^1$ and call the new function $\hat{\phi}$. Then, $\hat{\phi}(\boldsymbol{x}_1) \neq \hat{\phi}(\boldsymbol{x}_2)$.

If

$$\left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_1^1 - \boldsymbol{m}^1\right\|^2}{\sigma^2} \right\} \right]^{\gamma_1} \neq \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_2^1 - \boldsymbol{m}^1\right\|^2}{\sigma^2} \right\} \right]^{\gamma_1}$$

then select $\hat{\boldsymbol{m}}^1$ such that it is equidistant from both $\boldsymbol{s}_1^1$ and $\boldsymbol{s}_2^1$ so that $\hat{\phi}(\boldsymbol{x}_1) \neq \hat{\phi}(\boldsymbol{x}_2)$. Thus, Φ is a separating family.

Lemma 2: Φ contains the function 1.

Proof: A function $\phi_1 \in \Phi$ with $\gamma_j = 0$, $\forall j$ is the function 1.

Lemma 3: If $\phi \in \Phi$, then $\phi^n \in \Phi$ for any $n \in \Re$.

Proof: Consider

$$\phi(\boldsymbol{x}) = \prod_{j=i}^{l} \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_1^j - \boldsymbol{m}^j\right\|^2}{\sigma^2} \right\} \right]^{\gamma_j}$$

then

$$(\phi(\boldsymbol{x}))^n = \prod_{j=i}^{l} \left[\exp\left\{ -\frac{\left\|\boldsymbol{s}_1^j - \boldsymbol{m}^j\right\|^2}{\frac{\sigma^2}{n}} \right\} \right]^{\gamma_j}.$$

Hence, $(\phi(\boldsymbol{x}))^n \in \Phi$.

We will now prove the universal approximation property of GFSRBF using Stone–Wierstrass theorem.

Stone–Wierstrass Theorem [17]: Let X be a nonvoid compact set and G a separating family of functions on X containing the function 1; then for any continuous real-valued function

$f(\boldsymbol{x})$ defined on X, and for any $\epsilon > 0$, there exists a polynomial $p(\boldsymbol{x}) = \sum_{j=1}^{n} w_j (g_j(\boldsymbol{x}))^{n_j}$, where $g_j(x) \in G$ such that

$$\sup \{|f(\boldsymbol{x}) - p(\boldsymbol{x})| \ |\boldsymbol{x} \in X\} < \epsilon.$$

Let us denote the network output by $GFSRBF(\boldsymbol{x})$, for an $\boldsymbol{x} = (\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^l) \in \Re^p$. In other words

$$\text{GFSRBF}(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \prod_{j=1}^{l} \left[\exp \left\{ - \frac{\left\| \boldsymbol{s}^j - \boldsymbol{m}_i^j \right\|^2}{\sigma_i^2} \right\} \right]^{\gamma_j}.$$

1) Theorem: For any given continuous function f on the compact set $X \subset \Re^p$ and arbitrary $\epsilon > 0$

$$\sup \left\{ |f(\boldsymbol{x}) - \text{GFSRBF}(\boldsymbol{x})| \ | \boldsymbol{x} = (\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^l) \in X \right\} < \epsilon.$$

Proof: It directly follows from Lemmas 1–3 and the Stone–Wierstrass theorem.

Universal Approximation Property of GFSMLP

The universal approximation property of MLP with sigmoidal activation functions is well known [18]–[21], [30]. Here, we show that adding attenuation functions in the input layer does not affect the universal approximation property of an ordinary MLP.

For convenience, we assume one feature in each group. The net input to neuron j in the hidden layer is

$$\text{net}_j = \sum_i w_{ij} (x_i F_i) \tag{29}$$

where F_i is an attenuator of feature i. Note that if the groups contain more than one feature, then the attenuators for features in the same group will be the same. We can consider the attenuators as parts of the weights connecting the input to the first hidden layer. Thus, we can say

$$W_{ij} = w_{ij} F_i \tag{30}$$

and

$$\text{net}_j = \sum_i W_{ij} x_i \tag{31}$$

so the net input to a node in the first hidden layer remains of the same form as that of a conventional MLP. However, the weight W_{ij} is composed of two parts w_{ij} and F_i, where both w_{ij} and F_i are adjustable, w_{ij} is unrestricted in sign and magnitude, and $F_i \in [0, 1]$. For any trained MLP, if we consider W_{ij}'s to be the weights connecting the inputs and the nodes in the first hidden layer, then a decomposition as in (30) is always possible with the trivial choice of $F_i = 1, \forall i$. Thus, the GFSMLP is equivalent to an ordinary MLP, and hence the universal approximation property would be retained.

REFERENCES

[1] H. Al-Mubaid and N. Ghaffari, "Identifying the most significant genes from gene expression profiles for sample classification," in *Proc. IEEE Conf. Granural Comput.*, 2006, pp. 655–658.

[2] E. Anderson, "The irises of the Gaspe Peninsula," *Bull. Amer. IRIS Soc.*, vol. 59, pp. 2–5, 1935.

[3] L. M. Beleue and K. W. Bauer, "Determining input features for multilayered perceptron," *Neurocomputing*, vol. 7, no. 2, pp. 111–121, 1995.

[4] J. C. Bezdek, J. Keller, R. Krishnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Norwell, MA: Kluwer, 1999.

[5] C. Bhattacharyya, L. R. Grate, A. Rizki, D. Radisky, F. J. Molina, M. I. Jordan, M. J. Bissell, and I. S. Mian, "Simultaneous classification and relevant feature identification in high-dimensional spaces: Application to molecular profiling data," *Signal Process.*, vol. 83, pp. 729–743, 2003.

[6] C. L. Blake and C. J. Merz, "UCI Repository of Machine Learning Databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, 1998 [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository.html

[7] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, vol. 97, no. 1, pp. 245–271, 1997.

[8] F. Z. Brill, D. E. Brown, and W. N. Martin, "Fast genetic selection of features for neural network classifiers," *IEEE Trans. Neural Netw.*, vol. 3, no. 2, pp. 324–328, Mar. 1992.

[9] D. Chakraborty and N. R. Pal, "Integrated feature analysis and fuzzy rule-based system identification in a neuro-fuzzy paradigm," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 31, no. 3, pp. 391–400, Jun. 2001.

[10] D. Chakraborty and N. R. Pal, "A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 110–123, Jan. 2004.

[11] D. Chakraborty and N. R. Pal, "Two connectionist schemes for selecting groups of features (sensors)," in *Proc. 2003 IEEE Int. Conf. Fuzzy Syst.*, 2003, pp. 161–166.

[12] R. De, N. R. Pal, and S. K. Pal, "Feature analysis: Neural network and fuzzy set theoretic approaches," *Pattern Recognit.*, vol. 30, no. 10, pp. 1579–1590, 1997.

[13] A. P. Engelbrecht, "A new pruning heuristic based on variance analysis of sensitivity information," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1386–1399, Nov. 2001.

[14] K. Fukunaga, *Statistical Pattern Recognition*. New York: Academic, 1989.

[15] R. Haralick, K. Shanmugam, and I. Dinstein, "Texture features for image classification," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.

[16] E. J. Hartman, J. D. Keeler, and J. M. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximators," *Neural Computat.*, vol. 2, pp. 210–215, 1990.

[17] E. Hewitt, *Real and Abstract Analysis*. Berlin, Germany: Springer-Verlag, 1965.

[18] K. Hornik, "Some new results on neural network approximation," *Neural Netw.*, vol. 6, pp. 1069–1072, 1993.

[19] K. Hornik, "Approximation capabilities of multilayer perceptrons," *Neural Netw.*, vol. 4, pp. 251–257, 1991.

[20] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayered feedforward networks," *Neural Netw.*, vol. 3, pp. 551–560, 1990.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayered feedforward networks are universal approximators," *Neural Netw.*, vol. 2, pp. 259–366, 1989.

[22] A. Jain and D. Zongker, "Feature selection: Evaluation, application and small sample performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 148–153, Feb. 1997.

[23] I. T. Jollife, *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[24] D. Koller and M. Sahami, "Toward optimal feature selection," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 284–292.

[25] M. Kudo and J. Sklansky, "Comparison of algorithms that selects features for pattern classifiers," *Pattern Recognit.*, vol. 33, no. 1, pp. 25–41, 2000.

[26] A. S. Kumar, S. Chowdhury, and K. L. Mazumder, "Combination of neural and statistical approaches for classifying space-borne multispectral data," in *Proc. Int. Conf. Adv. Pattern Recognit. Digit. Tech.*, Calcutta, India, 1999, pp. 87–91.

[27] F. Kohavi and G. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, no. 1, pp. 273–342, 1997.

[28] A. Laha, N. R. Pal, and J. Das, "Designing prototype-based classifiers and their application to multispectral satellite images," in *Proc. 6th Int. Conf. Soft Comput.*, Japan, 2000, ISBN: 4-938717-04-2.

[29] M. Last, A. Kandel, and O. Maimon, "Information-theoretic algorithm for feature selection," *Pattern Recognit. Lett.*, vol. 22, pp. 799–811, 2001.

[30] M. Leshno, V. A. Lin, A. Pinkus, and S. Schocken, "Multilayered feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, pp. 861–867, 1993.
[31] Y. Lin and G. A. Cunningham, III, "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 2, pp. 190–198, May 1995.
[32] H. Liu and L. Yu, "Towards integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, May 2005.
[33] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, 1992.
[34] D. J. C. MacKay, "A practical Bayesian framework for backprop networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.
[35] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 5, pp. 698–714, 1992.
[36] R. M. Neal, "Bayesian learning for neural networks," in *Lecture Notes in Statistics*. Berlin, Germany: Springer-Verlag, vol. 118.
[37] J. Novovicova, P. Pudil, and J. Kittler, "Divergence based feature selection for multimodal class densities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, pp. 218–223, 1996.
[38] N. R. Pal, "Soft computing for feature analysis," *Fuzzy Sets Syst.*, vol. 103, pp. 201–221, 1999.
[39] N. R. Pal and K. K. Chintalapudi, "A connectionist system for feature selection," *Neural Parallel Sci. Comput.*, vol. 5, no. 3, pp. 359–381, 1997.
[40] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Comput.*, vol. 3, pp. 246–257, 1991.
[41] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 2, pp. 164–171, Apr. 2000.
[42] M. R. Rezaee, B. Goedhart, B. P. F. Lelieveldt, and J. H. C. Reiber, "Fuzzy feature selection," *Pattern Recognit.*, vol. 32, pp. 2011–2019, 1999.
[43] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayered perceptron," *J. Neural Netw. Comput.*, pp. 40–48, 1990.
[44] R. Setino, "Neural network feature selector," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 654–662, May 1997.
[45] J. M. Steppe, Jr., "Integrated feature and architecture selection," *IEEE Trans. Neural Netw.*, vol. 7, no. 4, pp. 1007–1014, Jul. 1996.
[46] P. Sykacek, "On input selection with reversible jump Markov chain Monte Carlo sampling," in *Advances in Neural Information Processing Systems*, S. A. Sola, T. K. Leen, and K. R. Müller, Eds. Cambridge, MA: MIT Press, 2000, vol. 12, pp. 638–644.
[47] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.
[48] M. Sugeno and T. Yasukawa, "A fuzzy-logic based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 7–31, Feb. 1993.
[49] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for detecting redundant inputs of perceptron networks," *Neurocomputing*, vol. 14, pp. 177–193, 1997.

Debrup Chakraborty received the B.E. degree in mechanical engineering from Jadavpur University, Kolkata, India, in 1997, and the M.Tech. and Ph.D. degrees in computer science from Indian Statistical Institute, Kolkata, India, in 1999 and 2005, respectively.

Currently he is with the Computer Science Department, Centro de Investigaciones y Estudios Avanzados del IPN, Mexico City, Mexico. His current research interests include design and analysis of provably secure symmetric encryption schemes, efficient software/hardware implementations of cryptographic primitives, pattern recognition, and neural networks.

Nikhil R. Pal (M'91–SM'00–F'05) received the B.Sc. degree in physics and the M.B.M. degree in operations research from the University of Calcutta, Calcutta, India, in 1978 and 1982, respectively, and the M.Tech. and Ph.D. degrees in computer science from the Indian Statistical Institute, Calcutta, India, in 1984 and 1991, respectively.

Currently, he is a Professor at the Electronics and Communication Sciences Unit, Indian Statistical Institute. He has coauthored a book titled *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing* (Norwell, MA: Kluwer, 1999) and edited/coedited several books. His current research interest includes image processing, pattern recognition, fuzzy sets theory, neural networks, evolutionary computation, and bioinformatics.

Dr. Pal serves on the editorial/advisory board of several journals including the *International Journal of Approximate Reasoning*, *International Journal of Hybrid Intelligent Systems*, *Neural Information Processing—Letters and Reviews*, *International Journal of Knowledge-Based Intelligent Engineering Systems*, *International Journal of Neural Systems*, and *Fuzzy Sets and Systems*. He is an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS—PART B: CYBERNETICS and the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He was the President and currently is 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 (1999, Calcutta, India) and was a Co-Program Chair of the 2005 and 2006 IEEE International Conference on Fuzzy Systems. He was the General Chair of the 2002 AFSS International Conference on Fuzzy Systems, Calcutta, India, and the 11th International Conference on Neural Information Processing (2004).