Two Connectionist Schemes for Selecting Groups of Features (Sensors)

Debrup Chakraborty
Electronics and Communication Sciences Unit
Indian Statistical Institute
Calcutta 700108, India
Email: debrup_r@isical.ac.in

Nikhil R. Pal
Electronics and Communication Sciences Unit
Indian Statistical Institute
Calcutta 700108, India
Email: nikhil@isical.ac.in

***Abstract*— Suppose for a given classification or function approximation (FA) problem data are collected using l sensors. From the output of the i^{th} sensor n_i features are extracted, thereby generating $p = \sum_{i=1}^{l} n_i$ features. So for the task at hand we have $X \subset R^p$ as input data along with their corresponding outputs or class labels. Here we propose two novel connectionist schemes that can select the sensors that yield redundant or bad features online and also do the required task, say, FA or classification. One of the schemes is based on the Radial Basis Function network and the other uses the Multilayered Perceptron network. Simulations show that the methods can detect the bad groups of features online and can also eliminate the effect of these bad features while doing the task.**

I. Introduction

It is known that for a given problem all features that characterize a data point may not be equally important, even some features may be *bad* features. Thus, selection of a proper subset of features from the available set of features is important for designing efficient classifiers or other function approximation (FA) systems. Methodologies for selecting good features on the basis of feature ranking etc. do exists [1], [5], [10], but most of these methods perform feature analysis in a separate phase, offline with the main learning process. In [5] and [7] it was noted that the goodness of a feature for a learning task depends on the learning algorithm itself. Hence, the feature subset which is appropriate for one learning algorithm may not be the best for other learning algorithms. Therefore, integration of the feature selection phase with the main learning task can yield better systems. Online feature selection techniques have been previously reported in [3], [4], [11].

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 now-a-days 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 emmision, 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. Conventional feature selection methods select good features from all available features generated from all these sensors. But, our objective here is to discard all the computed features from the signals obtained from redundant *sensors*, if any. The features computed from the output of a particular sensor can be taken as a group. This kind of grouping results in a natural partition of the total set of features according to their sensory origin. There may exist other natural groupings among features too. Here we address the problem of feature group selection, instead of individual feature selection. Of course, individual feature selection is a special case of this group feature selection methodology. *To our knowledge this problem has not been addressed in literature.*

In the rest of the paper we discuss two connectionist schemes to deal with this problem. The first scheme is a modified radial basis function network which we call as Group Feature Selecting Radial Basis Function (GFSRBF) network and the other is a modified multilayered perceptron called the Group Feature Selecting Multilayered Perceptron (GFSMLP).

II. The Group feature selecting Radial basis function (GFSRBF)network

If we have an input data set $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset R^p$ then a radial basis function network computes the function

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

where the ϕ_i's are the basis functions and $\mathbf{w}_i$s are parameter vectors known as weights. If we assume Gaussian type basis functions then

$$\phi_i(\mathbf{x}) = exp\left\{-\frac{||\mathbf{x} - \mu_i||^2}{\sigma_i^2}\right\}. \quad (1)$$

In eq. (1) μ_i and σ_i are the parameters related to the i^{th} basis function, commonly known as the center and spread respectively and $||.||$ is the Eucledian norm. Let us assume $\mathbf{x} = [x_1\ x_2\ ...\ x_p]^T$ and $\mu_i = [\mu_{i1}\ \mu_{i1}\ ...\ \mu_{ip}]^T$. Then we have

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

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

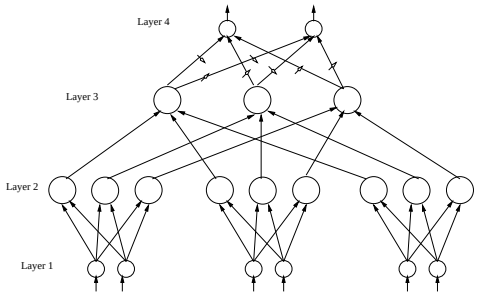

Fig. 1. The GFSRBF network structure

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

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

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

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

Thus, each C_i^j takes as 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/redundant* sensors. For the time being let us *pretend* that we know the bad or redundant groups. Hence, we aim to design the component functions in such a manner that a component function C_i^k corresponding to a bad/redundant group will always take the value of unity (1) irrespective of the input $\mathbf{s}^k$. If we can do so, then C_i^k will *never* contribute anything to the total process i.e., to $\phi_i(\mathbf{x})$. To achieve this we design the the component functions as:

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

Clearly, in eq. (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 $\mathbf{s}^j$. On the other hand, if $|\beta_j|$ is very large then C_i^j in eq. (5) reduces to C_i^j in eq. (4) resulting in no change of the role of the basis functions. But, how do we know which group is good and which is bad? 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 in mind we next discuss the network structure of GFSRBF.

A. The network structure

GFSRBF network is a four layer feed forward 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 $T = \{(\mathbf{x}, \mathbf{t})|\mathbf{x} \in R^p, \mathbf{t} \in R^c\}$. Each point $\mathbf{x}$ has p features which can be grouped into l groups. The division could be made based on sensors or some other criteria. In our subsequent discussions we denote the output of the i^{th} layer by z^i. We now discuss the general structure of the network layer by layer.

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 is called the *component function layer*, and this layer is responsible for the feature selection task. If the network contains m basis functions then this layer will contain $l \times m$ 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 i^{th} basis function and the j^{th} group - the superscript 2 denotes the layer number. Then we have

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

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

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

Layer 4 : This 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 is 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 i^{th} 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). But equation (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

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

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

B. The learning rules

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

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

Without loss we omit the subscript $\mathbf{x}$ and call the error term as E. We use the gradient descent technique to update the weights and the feature group modulators β_js. 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. Note that, along with w_{ij} and β_j, the parameters of the basis functions, i.e., the centers and the spreads could also be learnt using gradient descent technique which we do not do here, but we use judicious choices of them as discussed next.

C. Selection of centers and spreads

Generally there are two common strategies used in practice: (1) The parameters for the basis functions are chosen apriori and are kept fixed. Only the weights between the hidden and output layers get updated during learning. (2) All parameters are learnt by gradient descent (or a similar) technique. We follow the first strategy here. Initial centers and spreads are determined by the Fuzzy C Means clustering algorithm (FCM)[2]. We use the fuzzifier $m = 2$ in all reported results. Once we obtain the cluster centers μ_i by the FCM algorithm, we calculate the spread σ_i of the i^{th} basis function as $||\mu_i - \mu_j||$, where μ_j is the center of the basis function nearest to μ_i.

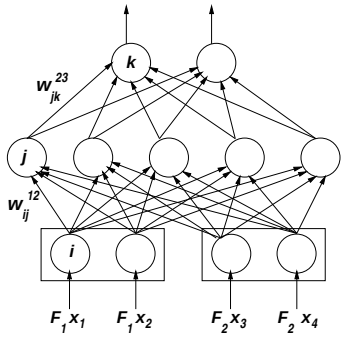

Fig. 2. MLP with group feature selection

III. A Group Feature Selecting Multilayered Perceptron

A multilayered perceptron (MLP) can also be modified to do group feature selection. A feature selecting MLP was proposed by Pal and Chintalapudi [11] which used sigmoidal attenuation functions in the input layer. We generalize their idea to propose GFSMLP, a modified form of MLP for group feature selection. Here the philosophy is different from that used for GFSRBF.

Let F_l be an attenuator function attached to the l^{th} group of features. Each feature x_i of the l^{th} group gets multiplied by the attenuator function F_l before it gets into the network. If $F_l = 0$, then no feature of the l^{th} group gets into the network, while if $F_l = 1$, then every feature of the l^{th} group enters the network unattenuated. For intermediate values of F_l, transformed values of the features enter into the network. Figure 2 shows the architecture of a group feature selecting MLP (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 $F_l \in [0, 1]$. There can be many choices for F_l, we take $F_l = e^{-\beta_l^2}$. β_l is a parameter related to the l^{th} group of features. If $\beta_l \to 0$, $F_l \to 1$ and if $\beta_l \to \pm\infty$, $F_l \to 0$. The objective here is to select appropriate values of β_ls through training such that $F_l \to 1$, if the l^{th} group is a useful group of features and $F_l \to 0$, if the l^{th} group is a *bad* or *redundant* group. The parameters β_l can be learnt by the backpropagation algorithm along with other parameters of a multilayered perceptron.

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

$$z_i^1 = x_i F_l. \tag{13}$$

Let w_{jk}^{23} be the weight of the link connecting the j^{th} node of the hidden layer with the k^{th} node of the output layer. Similarly, w_{ij}^{12} denotes the weight connecting the i^{th} node of

the input layer with the j^{th} node of the hidden layer. Thus,

$$z_j^2 = \frac{1}{1 + e^{-\sum_i z_i^1 w_{ij}^{12}}} \tag{14}$$

and

$$z_k^3 = \frac{1}{1 + e^{-\sum_j z_j^2 w_{jk}^{23}}}. \tag{15}$$

For an input $\mathbf{x}$ if the target is $\mathbf{t}$, then the error E is

$$E = \frac{1}{2} \sum_k (z_k^3 - t_k)^2. \tag{16}$$

We define

$$\delta_i^q = \frac{\partial E}{\partial z_i^q}, \qquad q = 1, 2, 3. \tag{17}$$

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

$$w_{jk}^{23}(t+1) = w_{jk}^{23}(t) - \eta \delta_k^3 z_j^2 z_k^3 (1 - z_k^3) \tag{18}$$

and

$$w_{ij}^{12}(t+1) = w_{ij}^{12}(t) - \eta z_j^2 (1 - z_j^2) z_i^1 \delta_j^2. \tag{19}$$

In the above equations, η is a predefined learning constant. The update equation for the feature attenuator of the l^{th} group, β_l, can be derived as

$$\beta_l(t+1) = \beta_l(t) + \mu \sum_{i \in S_l} 2\delta_i^1 z_i^1 \beta_l(t). \tag{20}$$

Here, μ is also a predefined learning constant. The β_ls 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_ls are changed in such a manner that for each important group $F_l \to 1$.

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.

IV. Experimental results

We provide here experimental results on two data sets: **Chem**, **RS-Data**.

The Chem data [12] set 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 inputs and one output. The input variables are monomer concentration (u_1), change of monomer concentration (u_2), monomer flow rate (u_3) and the two local temperatures inside the plant (u_4 and u_5). The only output (y) is the set point for monomer flow rate. In [12] there is a set of 70 data points obtained from an actual plant operation. It is known that u_4 and u_5 *do not significantly contribute* to the output [12].

RS-data set [6] is a 256 level seven channel satellite image data of size 512×512 pixels obtained from Landsat-TM3. The 512×512 ground truth data provide the actual distribution of classes of objects captured in the image. From these images we produce the labeled data set with each pixel represented by a 7-dimensional feature vector and a class label. Each dimension of a feature vector comes from one channel. This data have 8 classes representing different landcover types.

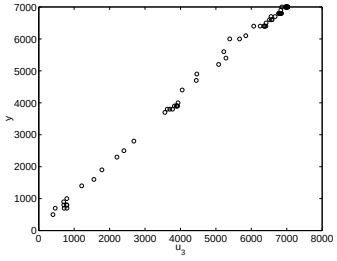

Fig. 3. Plot of y and u_3

TABLE I
Values of γ_j in GFSRBF for Chem data (considering 3 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

As stated earlier, the attenuation parameters for both networks are initialized with the aim, 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. And in GFSMLP we set $\beta_j = 3$ thus making $F_j = 0.0001, \forall j$. For both GFSRBF and GFSMLP the values of the feature attenuators and also the performance of the system depends on the initialization. For the data sets used we found that the features selected for various initializations were quite consistent.

A. Chem

In this example we demonstrate the feature selection capability of our network for the function approximation task. The five input features can be easily divided into 3 groups with respect to the type of features. The monomer concentration (u_1) and change of monomer concentration (u_2) can constitute one group, the monomer flow rate (u_3) as the second group and the temperature parameters (u_4 and u_5) can form the third group. With 15 basis functions we find that the network accepts only the second group of features, i.e., only u_3 is important for the task (see Table I). Figure 3 shows the plot of the output y with u_3. Figure 3 reveals a very strong correlation between u_3 and y(correlation coefficient = 0.9984!). Figure 4 depicts the performance of the network, i.e., compares the network output with the actual.

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

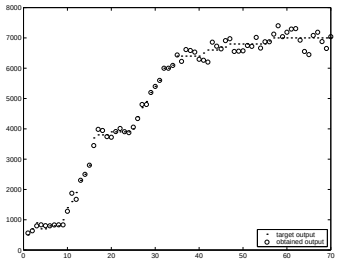

Fig. 4. Performance on Chem data

TABLE II
RANGE OF FEATURE VALUES FOR CHEM DATA

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

up u_3. But previously it has been reported that the features u_1 and u_2 also have some effect on the output [12]. Our network ignores that, and as a result we get reasonable (but not very good) performance as suggested by Fig. 4. This is neither a problem of the model nor 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 lies in [0,1]. We call this new data set a Normalized-Chem. With Normalized-Chem we run GFSRBF with 10 basis functions. The final values of the feature attenuators obtained for this network are shown in Table III. In this case we find that u_3 dominates, but the feature group $\{u_1, u_2\}$ also has a significant effect. The performance on this normalized data is shown in Figure 5, which exhibits a much better match. The Chem data set was used by Lin

TABLE III
VALUES OF γ_j IN GFSRBF FOR NORMALIZED CHEM DATA
(CONSIDERING 3 GROUPS OF FEATURES)

Group 1 $\{u_1, u_2\}$	Group 2 $\{u_3\}$	Group 3 $\{u_4, u_5\}$
0.0134	0.337	0.009

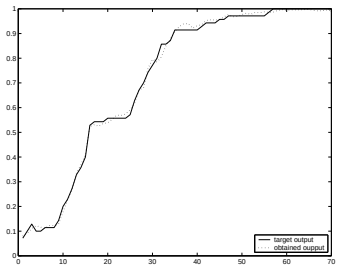

Fig. 5. Performance on normalized Chem data

TABLE IV
VALUES OF F_j IN GFSMLP FOR NORMALIZED CHEM DATA
(CONSIDERING 3 GROUPS OF FEATURES)

Group 1 $\{u_1, u_2\}$	Group 2 $\{u_3\}$	Group 3 $\{u_4, u_5\}$
0.98	1.00	0.00

and Cunningham in [9]. To evaluate a system they used a performance index PI defined as

$$PI = \frac{\sqrt{\sum_{k=1}^{n}(t_k - o_k)^2}}{\sum_{k=1}^{n}|t_k|}. \quad (21)$$

Here t_k and o^k are the desired and actual outputs respectively. They got a PI of 0.002245 on Chem by using features u_1, u_2, u_3. The PI for our method using Chem is 0.004271 and with Normalized-Chem is 0.002239.

With GFSMLP we obtained a PI of 0.00215 on Normalized-Chem data considering 3 groups of features as in the case of the GFSRBF network. The feature attenuator values for the three groups in Table IV show that the GFSMLP selects the same two important groups of features and completely rejects the third group. The PI value suggests that it can also do the function approximation job with a good accuracy. Note that GFSMLP gives relatively more importance on the first group than that by GFSRBF. This reemphasizes the fact that importance of a feature (or of a group of features) is a function of the tool being used to solve a problem.

B. RS-data

This data set contains 262144 points distributed in 8 classes. From each class we randomly select 200 points to get a training sample of 1600 points. For this data set initially we consider the 7 features present as 7 groups. Running our network 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 V, which reveals that the network completely discards the second feature. In [6] a misclassification of 21.8% was obtained on the test data. In [8], a misclassification of about 14% on the test data was reported. The performance of our system is comparable to theirs though our system considers only six features.

In another experiment we generated an additional feature from the each of the 7 channels of the image. For each pixel p we considered its 8-neighborhood and computed the standard deviation of the 9 pixels (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 along with d_p, as a feature. So we have 14 features divided into 7 groups. We call this as the RS14 data. Table VI shows the final values of the group modulators. From Table VI we can again conclude that the network rejects the features from the second sensor. This is consistent with the results described in Table V using 7 features. The misclassification obtained by this experiment is 20.6% on the training set and 16.5% on the test set.

TABLE V

VALUES OF γ_j IN GFSRBF FOR RS DATA (CONSIDERING 7 GROUPS, 1 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 VI

VALUES OF γ_j IN GFSRBF FOR RS DATA (CONSIDERING 7 GROUPS, 2 FEATURES PER GROUP)

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
0.99	0.00	0.92	0.99	0.46	0.99	0.99

TABLE VII

VALUES OF F_j IN GFSMLP FOR RS DATA (CONSIDERING 7 GROUPS, 2 FEATURES PER GROUP)

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
0.987	0.001	0.002	0.002	0.976	0.987	0.001

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 [2] on this data with all possible combination of 6 features, i.e. in each run we left out one feature. Among the 7 possible combinations, we obtained the least misclassification using the features 1,3,4,5,6 and 7. This clearly points out that feature 2 is a bad feature. This is quite consistent with the results obtained by us.

The GFSMLP selects only three groups of features from RS14 data. Table VII shows the final feature attenuator values. But, the GFSMLP gives a poorer classification of 23% on the training data and 21% on the test data. This should not be taken as a surprise or an indicator of poor performance of our system. If we retrain the system using the selected features, we are likely to get an improved performance. To demonstrate this we now use *only* the features selected by GFSMLP and trained an MLP with the Lavenberg-Marquardt optimization algorithm. In this experiment we obtained a misclassification of 13.2% on the training data and 14.6% on the test data using *only* the *three* groups of features (in total 6 features) selected by the GFSMLP. An ordinary MLP trained by the Lavenberg-Marquardt optimization algorithm with all the 14 features gives a misclassification of 7.44% on the training data and 12.28% on the test data. The misclassification obtained on the test data by using less than half the number of total avaiable features is quite comparable to that obtained by using all the features. So, the features selected by the GFSMLP are quite good, and the feature group that was rejected by the GFSRBF, is also rejected by GFSMLP.

V. CONCLUSION

Many real life applications use data from several sensors for decision making. Intelligent systems for automatic inspection and controlling of welding, medical diagnosis, 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 hazardous too (X-rays). Here, we like to work with the minimal number of sensors where each sensor could be responsible for a set of features. This is a more difficult problem than selecting individual features. In this paper we have proposed two novel schemes for the same one using the RBF network and the other using the MLP. The methods have been tested on two data sets and they exhibited excellent performance.

REFERENCES

[1] L.M. Beleue and K.W. Bauer, " Determining input features for multi-layered perceptron", *Neurocomputing*, vol 7, no 2, pp. 111-121, 1995

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

[3] D. Chakraborty and N.R. Pal, "Integrated feature analysis and fuzzy rule-based system identifi cation in a neuro-fuzzy paradigm", *IEEE Trans. on Systems Man Cybernetics B*, vol 31, no 3, pp. 391-400, 2001.

[4] D. Chakraborty and N.R. Pal, "Designing rule-based classifi ers with on-line feature selection: a neuro-fuzzy approach", *Advances in Soft Computing, LNAI 2275*, Springer, pp, 252-260, 2002.

[5] R. De, N.R. Pal and S.K. Pal, " Feature analysis: neural network and fuzzy set theoretic approaches", *Pattern Recognition* vol 30, no 10,pp. 1579-1590, 1997.

[6] A.S. Kumar, S. Chowdhury and K.L. Mazumder, "Combination of neural and statistical approaches for classifying space-borne multispectral data", *Proc. of ICAPRDT99, Calcutta, India*, pp. 87-91, 1999.

[7] F. Kohavi and G. John, "Wrappers for feature sebset selection", *Artificial intelligence*, vol 97, no 1, pp. 273-342, 1997.

[8] A. Laha and N. R. Pal, and J. Das, Designing prototype-based classifi ers and their application to multispectral satellite images, *Proc. 6th Int. conference on Soft Computing, IIZUKA2000*, Japan, 2000, CDROM ISBN 4-938717-04-2.

[9] Y. Lin, G.A. Cunningham III, "A new approach to fuzzy-neural system modeling", *IEEE Trans. Fuzzy Systems*, vol 3, no 2, 1995.

[10] N.R. Pal," Soft computing for feature analysis", *Fuzzy Sets and Systems*, 103, 201-221, 1999.

[11] N.R. Pal and K.K. Chintalapudi, "A connectionist system for feature selection", *Neural, Parallel & Scientific Computations*, vol 5. No. 3, 359-381, 1997.

[12] M. Sugeno and T.Yasukawa, " A Fuzzy-Logic based approach to qualitative modeling", *IEEE transactions Fuzzy Systems*, vol 1, no 1, pp. 7-31, 1993.