An Integrated Mechanism for Feature Selection and Fuzzy Rule Extraction for Classification

Yi-Cheng Chen, Nikhil R. Pal, *Fellow, IEEE*, and I-Fang Chung, *Member, IEEE*

***Abstract*—In our view, the most important characteristic of a fuzzy rule-based system is its readability, which is seriously affected by, among other things, the number of features used to design the rule base. Hence, for high-dimensional data, dimensionality reduction through feature selection (not extraction) is very important. Our objective, here, is not to find an optimal rule base for classification but to select a set of useful features that may solve the classification problem. For this, we present an integrated mechanism for simultaneous extraction of fuzzy rules and selection of useful features. Since the feature selection method is integrated into the rule base formation, our scheme can account for possible subtle nonlinear interaction between features, as well as that between features and the tool, and, consequently, can select a set of useful features for the classification job. We have tried our method on several commonly used datasets as well as on a synthetic dataset with dimension varying from 4 to 60. Using a ten-fold cross-validation setup, we have demonstrated the effectiveness of our method.**

***Index Terms*—Dimensionality reduction, feature modulators, feature selection, fuzzy rules.**

I. Introduction

FOR many application areas, given some input–output data, we need to find a model of the underlying system that transforms the input to the output. Such application areas include prediction of stock prices based on past observations, deciding on control action based on the present state of the plant, diagnosing cancers based on gene expression data, and many more. For each of these problems, we have $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$, where an unknown system S transforms $\mathbf{x}$ to $\mathbf{y}; \mathbf{y} = S(\mathbf{x})$. Here, $\mathbf{x} = (x_1, x_2, \ldots, x_p)' \in R^p$ is an input vector and $\mathbf{y} = (y_1, y_2, \ldots, y_r)' \in R^r$ is the corresponding output vector of a multiple-input multiple-output system. Our objective is to find an S' that approximates S to explain the given input–output data (X, Y). The popular tools that are used for such a problem are neural networks, fuzzy rule-based systems (FRBS), regression, support vector machines, etc. The success of such a system identification task depends strongly on the set of features that is used as input. This is true irrespective of the computational modeling tool that is used to identify the relation between the input and output. Contrary to the usual belief, more features are not necessarily good for system identification. Many features may lead to enhanced data acquisition time and cost, more design time, more decision making time, more hazards, more degrees of freedom (and, hence, higher chances of poor generalization), and more difficulty in identifying the system (local minima). Hence, reducing the dimensionality, if possible, is always desirable. The dimensionality reduction can broadly be done in two ways through feature selection [1]–[3] and feature extraction (computation of new features) [4], [5]. When feature extraction is used, the new features could be good to predict the output, but the new features may be difficult to interpret. This is clearly a disadvantage, particularly when the underlying applications are critical, such as diagnosis of cancer, prediction of blast vibration, and so on. Therefore, we consider only feature selection here.

Generally, feature selection methods are classified into two broad groups: Filter method and wrapper method [6]. The filter method does not require any feedback from the classifier or the predictor (function approximator) that will finally use the selected features. On the other hand, the wrapper method evaluates the goodness of the features using the classifier (or other prediction system) that will finally use the selected features. Obviously, wrapper methods are likely to yield better performance because the utility of a feature may also depend on the tool that is used to solve the problem. For example, the best set of features for a radial basis function network may not necessarily be the best set for a multilayer perceptron neural network [7]. The wrapper method although uses the predictor (induction algorithm) to assess the selected features, the optimal solution requires an exhaustive search considering all possible subsets of features, which is not feasible for high-dimensional datasets. Thus, typically, either a forward-selection or a backward-selection method, guided by some heuristic, is used [6]. Such methods treat the induction algorithm (the classifier here) as a black box. Since the feature selection is a step-wise process, it may fail to exploit the interaction between features. Here, we shall consider a different family of methods where the feature selection and identification of the required system are done *together* in an integrated manner. At the time of designing the predictor, it picks up the relevant features. Such a method can be called an *Embedded* method. Some advantages of such a method are: no need to evaluate all possible subsets (saves computation time), it can account for interaction between

Manuscript received January 11, 2011; revised August 22, 2011; accepted November 22, 2011. Date of publication December 26, 2011; date of current version August 1, 2012. This work was supported by the National Science Council, Taiwan, under Grant NSC-100-2221-E-010-011 and Grant NSC-100-2627-B-010-006.

Y.-C. Chen is with the Institute of Biomedical Informatics, National Yang-Ming University, Taipei 112, Taiwan (e-mail: weasley001@gmail.com).

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

I.-F. Chung is with the Institute of Biomedical Informatics, National Yang-Ming University, Taipei 112, Taiwan, and also with the Center for Systems and Synthetic Biology, National Yang-Ming University, Taipei 112, Taiwan (e-mail: ifchung@ym.edu.tw).

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/TFUZZ.2011.2181852

features, as well as that between the features and the tool that is used to solve the problem. There are several attempts at feature selection in this paradigm using neural networks [1], [7], genetic programming [8], support vector machines [9], [10]. Note that, for every model identification tool, it may not be easy to define such an integrated mechanism.

In [11], we have introduced an integrated mechanism to select useful features, while designing the rule base for function approximation/prediction type problem using the Takagi–Sugeno model. In this paper, we consider the classification problem. This is an integrated mechanism to select useful features and to extract fuzzy rules for classification. For the Takagi–Sugeno model, the consequents are functions of the input variables, while for the classifier model considered here, the consequents are class labels. Here, the firing strength (FS) of a rule determines the support of a rule for a class. Consequently, the system error whose minimization determines the set of selected features is different. In this context, we have suggested two different error functions. Although the basic feature selection philosophy is similar to that in [11], the rule structure, system error, and learning rules are quite different. Results of some preliminary investigation for cancer discrimination using only gene expression data based on this philosophy have been reported in [12].

In our view, one of the main advantages of an FRBS is its interpretability. Every atomic clause, such as $\mathbf{x}$ is CLOSE to 5.3 or $\mathbf{x}$ is Medium, can always be interpreted. However, if a rule has a large number of such atomic clauses (large number of inputs are involved), then it becomes extremely difficult for a human being to comprehend the relation represented by a rule. From this point of view, dimensionality reduction is even more important for an FRBS.

The literature on FRBS identification is quite rich [11], [13]–[46], and many articles have also been written for dimensionality reduction in the context of fuzzy rule extraction. Given a training dataset (X, Y), there are many ways to extract useful rules from this dataset. There are two main approaches to fuzzy rule extraction. One family of approaches uses a fixed partition of the input space to generate fuzzy rules, while the other family uses clustering.

The partition-based approaches [27]–[37] usually need a search algorithm to find the required rule set. For such methods, the search space increases exponentially with the dimensionality of the dataset and, usually, evolutionary algorithms are used for rule selection. Many of these methods implicitly realize feature selection at the rule level. In other words, a rule may not use a particular feature, and a feature can be dropped only if none of the rules uses that feature.

On the other hand, clustering-based approaches cluster the dataset and, then, translate each cluster into a fuzzy rule [11] [16]–[18], [20], [40], [41]. Rule extraction through clustering leads to many issues as explained in [11] and [51]. Here, we shall not consider these issues. We shall assume some ad hoc working solutions for these issues and primarily focus on feature selection aspect. The issue of finding useful features for system identification is related to structure identification. In this area as well, there have been many attempts. A number of these methods use hierarchical approaches [13], [14], [22]–[24] involving derived variables. Although such approaches are quite interesting, often the system loses its interpretability as a derived variable may not be easy to comprehend. In addition to these two main approaches, many authors use neuro-fuzzy systems for (implicit) extraction of fuzzy rules [2], [3], [42], [43].

To address the problem of integrated feature selection and rule extraction, in [2] and [3] using layered networks we proposed some neuro-fuzzy systems. The basic mechanism to select useful features is the same as that of the present one, but the neuro-fuzzy system suffers from conflicting rules that have to be eliminated by some postprocessing after the initial system is extracted. Another major disadvantage with such a neuro-fuzzy system is that even for a moderately high-dimensional data, the network size grows very rapidly that makes such a system less useful for many practical problems. In [11], we have introduced an integrated mechanism to select useful features, while designing the rule base for function approximation/prediction type problem using the Takagi–Sugeno model. This method eliminated the aforementioned problems. In this paper, we consider the classification problem.

II. Brief Survey of Some Existing Methods

First, we consider the family of methods that uses fixed fuzzy partition of feature space and selects a small useful set of fuzzy rules using some criterion. Often for rule selection, evolutionary algorithms are used. In this area, Ishibuchi and his colleagues [27]–[33] made the pioneering contributions. For example, in [27], Ishibuchi *et al.* start with a fixed fuzzy partition of the feature space, and then, the rule selection is formulated as a combinatorial optimization problem with two objectives: Minimization of the number of rules and maximization of the accuracy. Authors use a genetic algorithm to optimize a weighted combination of the two objectives.

In this paper, our objective is to discard bad (derogatory) features and indifferent features and select a set of features that can solve the problem satisfactorily. We decide the importance of a feature at the classifier level. However, there are methods which discard features at the rule level. For example, a feature F may be dropped by one rule but can be used by other rules. In such a case, the feature F is not a bad feature, but its importance to characterize different classes is different. For example, in [52], fuzzy rules are extracted from a decision tree and in this case different rules use different subsets of features.

Many of these partition-based approaches implicitly or explicitly select features at the rule level [27]–[35]. In [28], authors proposed an innovative method for fuzzy rule extraction from a fixed fuzzy partition of the input space. Because of the use of fixed fuzzy partition, it is easy to interpret the rules. Here, also the fuzzy rules are generated using a genetic algorithm. A very interesting aspect of this method is the use of don't care" fuzzy sets in the rule formation. The don't care" is found to have a very strong favorable impact on the performance of the rule

base. Note that the effect of "don't care" is like the elimination of features at the rule level.

Ishibuchi *et al.* [29] combined two fuzzy genetics-based machine learning (GBML) approaches (Michigan and Pittsburgh) into a single hybrid approach for fuzzy rule generation. This method exploits the benefits of both Michigan and Pittsburgh approaches. Authors also proposed a heuristic method for specification of antecedent fuzzy sets which is quite effective for high-dimensional datasets. This method is found to be more effective than the use of only Michigan or only Pittsburgh approach.

A steady-state genetic algorithm for extraction of fuzzy rules is proposed in [34] by Mansoori *et al.* This approach also starts with a given fuzzy partition of the feature space and considers the "don't care" linguistic value for each feature. This makes the search faster and also does feature elimination at the rule level. They used three different rule evaluation measures. For example, one of the rule evaluation measures is defined using a fuzzy version of the "support" to a fuzzy set as the average membership value of all data points to that fuzzy set.

Reducing the number of fuzzy rules and/or number of features, which enhances the interpretability of the system, usually has a negative effect on the accuracy of the rule base. A rule generation scheme thus faces a multiobjective optimization involving maximization of accuracy and minimization of the number of rules and/or minimization of the total length of the rule base (minimization of complexity) where these two objectives are conflicting in nature. There are many methods to deal with this accuracy–complexity/interpretability issue. For example, to search for a set of the nondominated solutions of the fuzzy rule selection problem, Ishibuchi *et al.* [30] proposed three kinds of fitness functions for a single-objective genetic algorithm and also provided a method based on a multiobjective algorithm.

An interesting scheme for comprehensible rule generation using a three-objective optimization problem is proposed in [31]. Here, the authors attempted to maximize the classifier performance, minimize the number of fuzzy rules and total number of antecedent clauses in the antecedent of the fuzzy rules. To find nondominated rule sets, authors used a modified multiobjective genetic algorithm (MOGA) and a hybrid fuzzy GBML.

Ishibuchi and Yamamoto [32] further proposed a two-step procedure to deal with three-objective optimization problems. The first step utilizes two rule evaluation measures, confidence and support of association rules, as prescreening criteria for fuzzy rule selection. A modified MOGA with a local search ability is proposed in the second step to enhance the searching efficiency for identification of a set of the nondominated solutions.

Ishibuchi and Nojima in [33] also explored the interpretability–accuracy tradeoff for fuzzy rule-based classifiers using three-objective optimization problems based on their GBML algorithm. Based on different considerations they adopted three different formulations for each of multiobjective and single-objective optimization problems. Based on the performance on the training data, they reported that a clear tradeoff structure can be visualized for each dataset that they have considered. However, because of possible overfitting during the training, a clear tradeoff structure may not be always obtained for test patterns.

Jin [38] first identified an initial rule base using a training dataset, which is then simplified based on a fuzzy similarity measure. The structure and parameters of the fuzzy system are optimized using genetic algorithms and gradient-based method. This results in a more interpretable fuzzy system. Here, different rules may involve different number of input variables. This method may not eliminate any input variable unless that variable is not involved in any rule. In [39], authors use ant colony optimization to select features minimizing the classification error. They use the Takagi–Sugeno model for classification.

Gacto *et al.* [35] presented a multiobjective evolutionary algorithm for fuzzy rule generation for regression problem. Although this paper is not about classification, it addresses the accuracy and complexity tradeoff issue nicely, and hence, we make a brief discussion of this approach. Interpretability can be viewed at different levels. For example, high-level interpretability can be enhanced by reducing the number of rules, number of linguistic variables, number of linguistic values, and by reducing the average length of a rule. While the low-level interpretability can be enhanced by tuning the membership functions with a view to minimizing the system error maintaining semantic integrity of the memberships. In this context, authors proposed a semantic-based interpretability index called Geometric Mean of 3 Metrics (GM3M), which is the product of three components. Then, authors used a multiobjective evolutionary algorithm that performs selection of fuzzy rules along with tuning of the membership functions in order to improve the system's accuracy as the first objective. The method also considers the model complexity as the second objective and the GM3M index, which attempts to preserve the semantic interpretability, as the third objective.

Next, we consider some methods that explicitly do dimensionality reduction (either through selection or extraction) for fuzzy rule generation. One of the pioneering works for fuzzy rule extraction using clustering and feature selection is due to Sugeno and Yasukawa [40]. Here, the authors used an iterative method to select the important features in an incremental manner. First, fuzzy models are generated with each input. Then, a regularity criterion is used to evaluate each such model. The input feature that is associated with the model having the best value of regularity index is taken as the most important variable. Then, the selected feature is combined with each of the remaining features and the best model, in terms of the index, is selected. The process is continued till the regularity criterion starts increasing its value.

The concept of hierarchical fuzzy system has been used by several authors [13], [14]. At the first level of the hierarchy, a set of important features is used to design a system that produces an approximate output of the system. Then, they use another set of useful features as well as the estimated output of the previous (first) level as inputs to obtain the rules for the next

level. This approach can avoid the rule explosion problem, but such a system is difficult to comprehend as it uses output of the previous level as input to the next level.

To realize a better readability, in [22] and [23], the output of the previous layer as input was not used. However, they used the output of the previous layer in the consequent part of the rules in the next layer. Joo and Lee [24], on the other hand, used the same basic hierarchical structure used in [22] and [23] but the consequents of the rules consist of only outputs of the fuzzy logic units (FLUs) in the previous layer. The antecedent of a rule uses only input variables and the consequent involves a weighted sum of products of all possible pairs of outputs from the FLUs in the previous layer in addition to a weighted sum of the FLU outputs. This makes it difficult to interpret the rule base. More importantly, this method also does not eliminate any input feature. In [44], Pomares *et al.* proposed a scheme for structure determination that can select the required input features as well as determine the number of membership functions on each selected feature.

Simplification of fuzzy rules through an explicit feature elimination step is also attempted [45]. Here, a measure of fuzzy entropy is used in conjunction with the backward elimination method to select useful features. Depending on the classifier error rate, the feature elimination process is stopped. This being an incremental process, it may not account for possible nonlinear interaction that may be present between features.

Sanchez *et al.* [46] defined mutual information for fuzzy random variables. In conventional approaches to feature selection using mutual information, dependence between variables is computed before they are fuzzified. However, according to these authors, the dependence may be influenced by the shape of the membership functions defined on them, and hence, they defined mutual information for fuzzy random variables and used that for feature selection for fuzzy rules [46]. The method proposed in [47] is guided by the fact that poor features should not have much influence on the performance of a fuzzy system. Here, authors defined an interesting measure of information contained in a fuzzy model. This measure is used to define the influence of a feature on the performance of the system.

In the recent past, there have been many attempts to do feature selection using fuzzy-rough approaches also [48]–[50]. Margin maximization and fuzzy concepts have also been combined to develop feature selection algorithms [53]. However, these methods do not use a fuzzy rule-based classifier.

III. Integrated Feature Selection and Rule Extraction

Let us consider a c-class problem. We are given a training dataset, $X = X_1 \cup X_2 \cup \ldots \cup X_c,\ X_i \cap X_j = \Phi,\ i = 1, \ldots, c; |X| = n$. Here, X_j is the training data from class j, $\mathbf{x} = (x_1, x_2, \ldots, x_p)' \in R^p$ is the input vector for an object, and c is the number of classes. There are different types of classification rules [54]. In this investigation, we shall restrict ourselves only to rules of the form: If x_1 is LOW and $\ldots$ and x_p is HIGH, then class is j, $j \in \{1, 2, \ldots, c\}$.

Each class will be represented by a set of rules. Let n_j be the number of rules describing the *j*th class. In general, the *i*th rule of the *j*th class has the form R_{ji}: If x_1 is $A_{1,ji}$ and $\ldots$ and x_p is $A_{p,ji}$, then class is j, $j \in \{1, 2, \ldots, c\},\ i = 1, \ldots, n_j$. Here, $A_{k,ji}$ is a linguistic value (fuzzy set) defined on the *k*th feature (linguistic variable) for the *i*th rule of the *j*th class. The total number of rules is $N_R = \sum_{j=1}^{c} n_j$. For a given feature value x_k for the *k*th feature, the membership value associated with the *i*th rule of the *j*th class will be denoted by $\mu_{k,ji}$.

The FS of a rule is computed using a T-norm. For a given object $\mathbf{x}$, suppose α_{ji} is the FS of the *i*th rule for the *j*th class. Let $lk = \text{argmax}_{ji}\{\alpha_{ji}\}$. In this case, since the FS of the *l*th rule representing the *k*th class is the highest, the associated object is assigned to class k.

Our objective, here, is to use good features and eliminate the features that have poor discriminating power or that can confuse the learning process. For any such system, features can be categorized into four groups [7]: 1) essential features, which are necessary irrespective of the modeling tool that we use; 2) bad or derogatory features, which must be dropped; 3) indifferent features that neither help nor cause problem in discrimination; and 4) redundant features, these are the features which are dependent; for example, two correlated features. Our primary objective is to select necessary features and discard bad features. We should also try to discard indifferent features and minimize the use of redundant features.

In a rule-based framework, if we want to eliminate the effect of a bad feature, we need a mechanism so that there is no effect of this feature on the FS of a rule, irrespective of the definitions of the fuzzy sets, the value of the variable, and the T-norm used. As done in [2], [3], [11] and [12], we use the concept of feature modulator. The feature modulator, conceptually, acts like a gate which prevents a bad feature to influence the FS of a rule involving that feature. The FS of a rule is computed using a T-norm [55]. One of the properties of a T-norm T is that "$T(1, x) = x$." Thus, if we can devise a mechanism so that it modulates the membership value corresponding to a poor feature to 1, irrespective of the fuzzy set and value involved, then the effect of the poor feature will be eliminated from the rule. That is what we do here.

A. Choice of Modulator (Gate) Functions

In order to realize the proposed philosophy, we use exactly one modulator or gate function for each feature. Since there are p features, there will be p modulators. The number of modulators neither depends on the number of rules nor on the number of fuzzy sets (linguistic values) defined. Each modulator has a tunable parameter λ that is used to model the extent a gate is opened (i.e., the extent a feature is useful). A simple choice for a modulator could be $G(\mu, \lambda) = \mu' = \mu^{M(\lambda)}$, where $M(\lambda) = \frac{1}{1+\exp(-\lambda)}$. The scalar variable λ is called the modulator parameter. To make it clear, let $\mu_{k,ji}$ be a membership value associated with the *i*th rule of the *j*th class for the *k*th feature.

Then, the modulated membership value $\mu'_{k,ji}$ is computed as

$$\mu'_{k,ji} = (\mu_{k,ji})^{\frac{1}{1+\exp(-\lambda_k)}}. \quad (1)$$

Now, in (1), if $\lambda_k \approx +\infty$ (actually a moderately high value is enough), then $\mu_{k,ji} \approx \mu_{k,ji}{}'$. However, if the magnitude of $\lambda_k \approx -\infty$ (a negative high value), then $\mu_{k,ji} \approx 1$, i.e., the modulated membership value becomes $\approx$1, irrespective of the actual membership value.

Note that several other choices of the M-function are possible. Another such modulator function is

$$M1(\lambda) = \exp(-\lambda^2). \quad (2)$$

Using (2), the modified membership value would be $G(\mu, \lambda) = \mu' = \mu^{M1(\lambda)}$. In other words

$$\mu'_{k,ji} = (\mu_{k,ji})^{\exp(-(\lambda_k)^2)}. \quad (3)$$

In this case, if $\lambda_k \approx 0$, then $\mu_{k,ji} \approx \mu_{k,ji}{}'$, i.e., the modulated membership values remain practically the same. On the other hand, when the magnitude of λ_k is high, then $\mu'_{k,ji} \approx 1$, i.e., the modulated membership value is nearly 1 irrespective of the actual membership value. In this investigation, we use formulation (3).

In order to find useful features, we need to devise a mechanism to set the values of λs to a high value (high in magnitude) for poor features and low values (nearly zero, low in magnitude) for useful features. We shall achieve this through a learning algorithm using some training data.

B. Choice of Error Function

So far, we did not mention about the specific T-norm that will be used to compute the FS. Typically, product and minimum are used. Here, we use product because it is differentiable.

Thus, using product, the modulated FS α_{ji} of the *i*th rule of the *j*th class will be

$$\alpha_{ji} = \Pi_{k=1}^{p} (\mu_{k,ji})^{\exp(-(\lambda_k)^2)}. \quad (4)$$

For learning the modulator parameters, we minimize a measure of system error. One possible choice of the error is [18]

$$E^X = \sum_{x \in X} E_x = \sum_{x \in X} (1 - \alpha_c + \alpha_{\neg c})^2. \quad (5)$$

In (5), $\boldsymbol{x} \in X$ is from the class c and R_c is the rule from class c giving the maximum FS α_c for $\boldsymbol{x}$. In addition, $R_{\neg c}$ is the rule with the maximum FS $\alpha_{\neg c}$ for $\boldsymbol{x}$ considering all rules from the remaining $c-1$ incorrect classes. This error function is found to be quite effective in many applications [56]. In place of (5), the more conventional error function (6) can also be used

$$E^X = \sum_{x \in X} E_x = \sum_{x \in X} \sum_{l=1}^{c} (o_l - t_l)^2. \quad (6)$$

In (6), o_l is the support for the *l*th class by the rule base and is defined as $o_l = \max_i\{\alpha_{li}\}; l = 1, \ldots, c$, and t_l is the target output for class l. Typically, if the data point is from class k, then $t_l = 1$ for $l = k$ and $t_l = 0 \,\forall\, l \neq k$. Note that such an error function allows one to use fuzzy/possibilistic label vectors also, if available. Here, we use (6). However, directly we cannot use (6) as it involves the "max" operator, which is not differentiable and we want to use gradient-based search algorithm to find the desirable values of the modulator parameters λ. Therefore, we use a softer version of maximum "softmax" that is differentiable. There are different choices for "softmax." We use the following formulation:

$$\text{softmax}(x_1, x_2, \ldots, x_p, q) = \frac{\sum_{i=1}^{p} x_i e^{q x_i}}{\sum_{i=1}^{p} e^{q x_i}}; \quad x_i \in [0, 1]. \quad (7)$$

As $q \to \infty$, softmax tends to the maximum of all x_i's, $i = 1, 2, .., p$. Based on computational experience, we find that a q around 50 can realize a good approximation to the maximum. Here, we use $q = 50$ in all our experiments.

The modulator learning algorithm can be realized either in online mode or in batch mode. To avoid the dependence of the results on the order of the data feeding, we use the batch version of the learning algorithm. Thus, for each input data, the corrections to all modulators are aggregated and then updated once for each pass through the data.

The basic philosophy of learning is that we assume *all features as unimportant* (or bad) at the *beginning* of the training. Then, we minimize the system error defined previously to learn the modulators. To make all features unimportant, we randomly initialize the modulators so that all modulated membership values are almost one (1) at the beginning of the training. This is realized by setting the initial value of λ as 3 + random Gaussian noise with mean zero and standard deviation 0.2. Using the java randGaussian() function, we generate the initial λs as $\lambda_k = 3 + \text{randGaussian}() \times 0.2 \,\forall\, k$.

At this point, a natural question arises: Can we initialize the modulators to low values so that at the onset of training every feature is important and then discard the bad features through training? The answer is No! Our objective is to select good features and eliminate bad (derogatory) features and indifferent features through the learning process. Our learning mechanism is based on gradient descent-based search algorithm, and hence, we must pretend that all features are bad features at the beginning of training. If we keep all gates open (low modulator values) at the beginning, then we cannot discard indifferent features because an indifferent feature neither will cause any problem nor will help, and hence, they will always be selected by the system. Moreover, if we keep the gates for dependent features open at the beginning of the training and if those features have some discriminating power, they will remain open as they do not increase the error in the system. Hence, in our framework, the only effective approach is to initialize the modulators so that each gate is almost closed at the beginning of the training.

C. Initial Rule Base

The modulator learning algorithm works on an initial rule base. For this, we cluster the training data from the *j*th class, $X_j \subseteq R^p$, into n_j clusters. Any clustering algorithm, such as the hard c-means, mountain clustering algorithms [16], [55],

can be used. We use the k-means [55] clustering algorithm. Since the main objective of this investigation is simultaneous feature selection and rule extraction, we do not address the issue of choice of optimal number of rules for a class. Instead, we assume a fixed number of rules for a class and demonstrate the effectiveness of our scheme.

Each cluster is expected to represent a dense area in the input space that is represented by the associated cluster centroid. We convert each such cluster into a fuzzy rule. For example, if the center of the *i*th cluster in X_j is $\boldsymbol{v}_i \in R^p$, then this cluster is converted into the rule

R_i: if $\boldsymbol{x}$ is CLOSE TO $\boldsymbol{v}_i$, then the class is j.

The fuzzy set "CLOSE TO" can be modeled by a multidimensional membership function such as

$$\mu_{\text{CLOSETO}\,\mathbf{v}_i}(\mathbf{x}) = \exp^{-\frac{\|\mathbf{x}-\mathbf{v}_i\|^2}{{\sigma_i}^2}}$$

where $\sigma_i > 0$ is a constant. Such a multidimensional membership function is difficult to interpret (less readability) and may not perform always well, particularly when different features have considerably different variances. Thus, such a rule is expanded as a conjunction of p atomic clauses

R_i: If x_1 is CLOSE TO v_{i1} AND ... AND x_p is CLOSE TO v_{ip}, then class is j.

We note that the earlier two forms are not necessarily the same.

D. Choice of Membership Function

The next issue is the choice of the membership function, CLOSE TO v_{ik}. Here, again, different choices such as triangular, trapezoidal, and Gaussian are possible. We use the Gaussian membership function because it is differentiable, which can help tuning of the membership parameters, if we want to. To be more explicit, CLOSE TO v_{ik} is modeled by

$$\mu_{ik}(x_k; v_{ik}, \sigma_{ik}) = \exp(-(x_k - v_{ik})^2/{\sigma_{ik}}^2).$$

The spread of the membership function can be initialized in several ways. For example, we can use the standard deviation of the *k*th component of the training data that are included in the associated cluster. The spread can also be initialized to a fixed value when every feature is normalized using Z-score, i.e., the normalized feature value x is obtained as $x' = (x - \text{Mean}(x))/\text{StandardDeviation}(x)$. In either case, these parameters can be further tuned using the training data. In this investigation, since for every feature, we use the Z-score normalization, every feature has a unit variance after normalization, and hence, we use $\sigma_{ik} = 1\,\forall i, k$ to guarantee substantial overlap between adjacent membership functions defined over a linguistic variable and we do not tune them. In addition, to see the effect of the choice of fixed spread for the membership functions, we have further performed some additional experiments using one dataset with different σ values. The details of these experiments are discussed in Section IV.

Next, we present the modulator learning algorithm.

Modulator Learning Algorithm

Begin
Choose: learning parameter, η_m; a parameter reduction factor, $0 < \varepsilon < 1$; maximum number of iterations, *maxiter*; initial $\lambda_k^0; k = 1, \cdots, p$.

Compute the error E_0^X and the number of misclassifications M_0^X for the initial rule base R^0.

$t \leftarrow 1$

While ($t \leq maxiter$) do

$$\lambda_k^t \;=\; \lambda_k^{t-1} - \eta_m \frac{\partial E_{t-1}^X}{\partial \lambda_k^{t-1}}; \forall k = 1, \cdots, p$$

Compute the error E_t^X for the new rule base R^t using (6).
If $E_t^X > E_{t-1}^X$
then
$\eta_m \leftarrow (1 - \varepsilon)\eta_m$
$R^t \leftarrow R^{t-1}$
/* Since the error is increased, we reduce the learning coefficient and restore the old modulator values. */
If $M_t^X = 0$ or $E_t^X \approx 0$
then Stop
$t \leftarrow t + 1$
End while
End

Note that in the aforementioned algorithm, we do *not* tune the center and spread of the associated membership functions. One may be tempted to do so adding two more update equations for the centers and spreads in the while loop. However, we do not make simultaneous tuning of membership parameters (centers and spreads) and the feature modulators because membership tuning has a local effect (it considers performance of each rule), while modulators have a global effect on all rules (it considers all rules involving a particular variable). Tuning of membership parameters, if needed, is recommended to be done in a separate phase after the learning of modulators is over and the rule base is simplified. We have not tuned the parameters of membership functions because even without that, for every dataset, we could get acceptable accuracy, and finding of optimally tuned rules is not our objective.

We also want to note that this method uses gradient descent. Hence, if a feature has some discriminating power and the system training error can be further reduced by that feature, the gate associated with that feature may slowly open with iteration even when an adequate number of useful features are selected by the system. That is why we recommend that *for feature selection, the modulator learning should be stopped when the training error (or misclassification) reaches an acceptable level*.

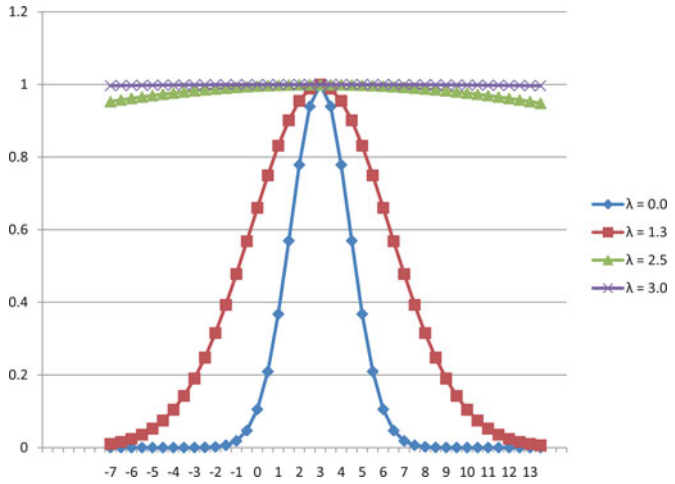


Fig. 1. Effect of λ on the modulated membership function. The original membership function (blue line with symbol "diamond," $\lambda = 0$) corresponds to $m = 3$ and $\sigma = 2$.

E. Choosing a Threshold for the Modulators

In order to select a set of useful features, we need to choose a value for the threshold T. Here, we provide a judicious way to choose the threshold when the membership functions are Gaussian [3]. Let us first consider the effect of the modulator on the modulated membership function.

The modulated membership value for a Gaussian membership function becomes

$$\mu' = \mu^{e^{-\lambda^2}} = (e^{\frac{-(x-m)^2}{\sigma^2}})^{e^{-\lambda^2}}.$$

Let $\gamma = e^{-\lambda^2}$. Then, the modulated membership value takes the form

$$\mu' = e^{\frac{-(x-m)^2}{\sigma'^2}}$$

where $\sigma' = \frac{\sigma}{\sqrt{\gamma}}$. Thus, the modulated membership function is also a Gaussian function, but with a modified spread that makes the membership function flat (membership values close to 1 over a wide range) for a high value of λ. In Fig. 1, we show the modulated membership functions for different values λ. The original membership function that is shown in this figure (blue line with symbol "diamond," $\lambda = 0$) corresponds to $m = 3.0$ and $\sigma = 2.0$. Fig. 1 depicts that, for $\lambda = 3$, the modulated membership values for x at $m \pm 2\sigma$ are practically one (1). Note that, at $x = m \pm 2\sigma$, the unattenuated membership value is 0.018 only. For a useful feature, λ will be reduced, and the modulated membership function will move toward the original membership function. If for a feature, at $x = m \pm 2\sigma$, the modulated membership value comes down from 1 to 0.5 (say), then we can assume that the feature is important. This idea can be used to choose a threshold. Thus, setting $\mu' = 0.5$ at $x = m \pm 2\sigma$, we get $\lambda = 1.324$. Therefore, a feature may be considered good, if the associated $\lambda < 1.324$. In particular, here, we use the threshold $\lambda = 1.3$. We also show results for a few other choices of the threshold.

IV. Results

A. Dataset

To validate our method, we have designed a synthetic dataset. The detailed information for the synthetic dataset is given next.

Synthetic: This is a four-class dataset in four dimension. There are 50 points in each class as depicted in Fig. 2(a), which is a scatter plot of the first two features F1 and F2. Features F1 and F2 are the good features defining the structure of the classes. We have added two noisy features as random values in [0, 10] and thereby making it a 4-D dataset. Fig. 2(b)–2(f) depict, respectively, the scatter plots of F1 and F3, F1 and F4, F2 and F3, F2 and F4, and F3 and F4.

In addition, we have used seven real datasets: Iris, Wisconsin Diagnostic Breast Cancer (WDBC), wine, heart, ionosphere, sonar, and glass, which are available from the University of California, Irvine, machine learning repository [57]. We have applied some preprocessing steps on some of these datasets: 1) For the heart data, we have considered only six real-valued features; and 2) every feature in each dataset is normalized using Z-score normalization process. These datasets are summarized in Table I , which includes the number of classes, number of features, and the size of the datasets including the distribution of data points in different classes. Here, we note, that for the first five datasets (synthetic, iris, WDBC, wine, and heart), we use only two rules for every class, while for the last three datasets (ionosphere, sonar, and glass) with more complex class structures, we use three rules for each class. It is worth noting, here, that the glass dataset has a significant overlap between different classes. To reveal this, we have used a scatter plot (see Fig. 3) of the top two principal components.

B. Experimental Protocol

In order to show the effectiveness of the features selected by our method, we use a ten-fold cross-validation framework as explained in the flowchart in Fig. 4. The k-means algorithm converges to a local minimum of its associated objective function [55] and the local minimum depends on the initial condition. Consequently, different initializations may lead to different sets of cluster centroids and, hence, different sets of initial rules. If a dataset has redundant features, then different initial rule base may lead to different sets of selected features. If the total system error is at an acceptable level in each of these runs, then each of these sets of features is equally effective for the discrimination task at hand. Moreover, the training-test partition may also influence the set of selected features. In order to minimize the effect of such dependence, we use a ten-fold cross-validation mechanism and the cross-validation experiment is repeated a large number of times (in this investigation, it is repeated 30 times). The entire procedure is outlined in Fig. 4.

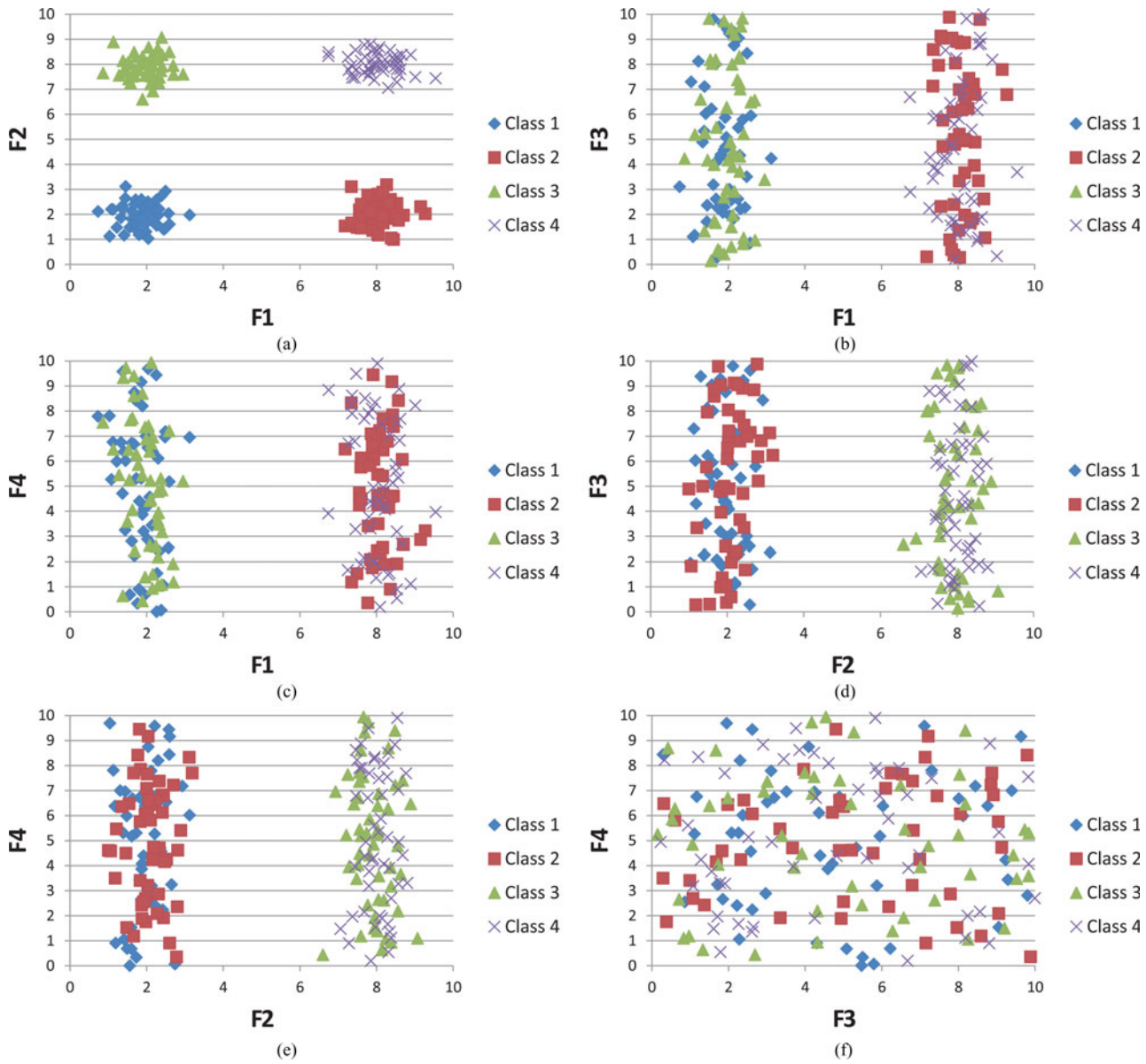


Fig. 2. Scatter plot of pair-wise features of the synthetic data. (a) Features F1 and F2. (b) Features F1 and F3. (c) Features F1 and F4. (d) Features F2 and F3. (e) Features F2 and F4. (f) Features F3 and F4.

TABLE I
SUMMARY OF THE DATASETS USED

Data Set	#Classes	#Features	Size of Data Set (distribution over classes)
Synthetic	4	4	200 (50+50+50+50)
Iris	3	4	150 (50+50+50)
WDBC	2	30	569 (357+212)
Wine	3	13	178 (59+72+47)
Heart	2	6	270 (150+120)
Ionosphere	2	33	351 (225+126)
Sonar	2	60	208 (97+111)
Glass	6	9	214 (70+76+17+13+9+29)

Given the training data $S = (X, Y)$, we randomly divide the members of X (along with their associated $\mathbf{y}$s) into m subsets for an m-fold cross-validation experiment (here, we perform this data subdivision randomly without enforcing class balance)

$$X = \bigcup_{i=1}^{m} \mathcal{X}_i$$

such that $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset \ \forall \ i \neq j$.

Then, we use one of the subsets, $\mathcal{X}_k$ as the test data, X_{TE} and the union of the remaining subsets as the training data $X_{TR} = \bigcup_{i=1, i\neq k}^{m} \mathcal{X}_i, \quad i \neq k$. Now, we divide X_{TR} into c subsets, $X_{TR,i}, \ i = 1, \ldots, c$ such that $X_{TR} = \bigcup_{i=1}^{c} X_{TR,i}, X_{TR,i} \cap X_{TR,j} = \emptyset \ \forall \ i \neq j$, where $X_{TR,i}$ contains all data points from the class i. Next, we cluster $X_{TR,i}$ into n_i number of clusters. Each such cluster is then converted into a fuzzy rule. With this rule base, we learn the feature modulators using the training data X_{TR}. After learning of the modulators, features with modulator values less than a threshold T are selected. Let

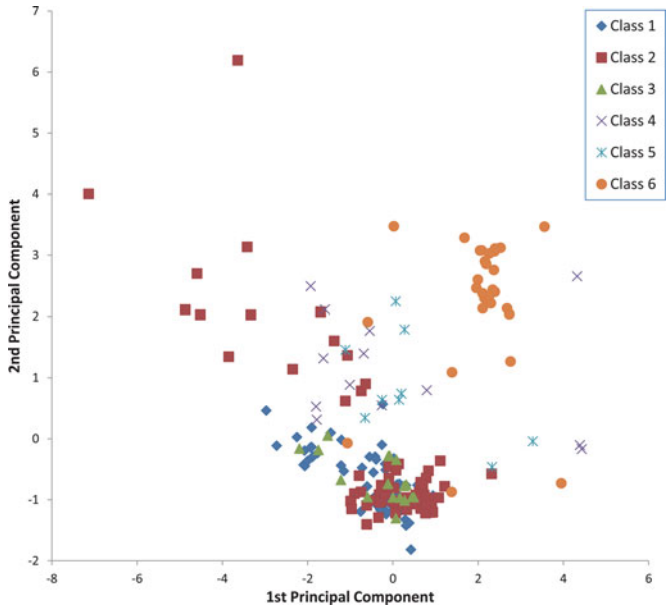


Fig. 3. Scatter plot of the top two principal components of the normalized glass data.

the number of selected features be NF_k. Using this set of selected features, we cluster $X_{TR,i}$ into n_i clusters to generate a set of n_i rules in the reduced dimension. This is repeated for each $X_{TR,i}$, $i = 1, \ldots, c$. In this way, we get a set of $N_R = \sum_{i=1}^{c} n_i$ rules in the reduced dimension. This rule set is tested on the test data X_{TE} considering only the selected features. Let eTE_k be the test error using $X_{TE} = X_k$ as the test data. The process is repeated for each of the ten folds, $k = 1, \ldots, 10$. Then, $eTot = \sum_k eTE_k$ is the total error from one such ten-fold cross-validation experiment. We also compute the average number of selected features as $N_F = \sum_k NF_k/10$.

The entire procedure is, then, repeated K times (here, $K = 30$). We report the average and standard deviation of $eTot$ and N_F. In order to further demonstrate how effective the feature selection scheme is, we also compute the ten-fold cross-validation test error using all features.

C. Results

Table II summarizes the results for all eight datasets that are considered in this study. In column one, we include the dimension of the dataset in parentheses. In this table, we also report the performance for different choices of threshold T. The last column of this table shows the average misclassifications considering all features. Note that instead of applying different thresholds on the same experimental results to select features, we have conducted separate experiments for each threshold. That is why for each dataset, we have three results, even when we consider all features. These results further demonstrate that the randomness that is associated with initialization of the clustering algorithm does not have a noticeable effect on the performance of our algorithm.

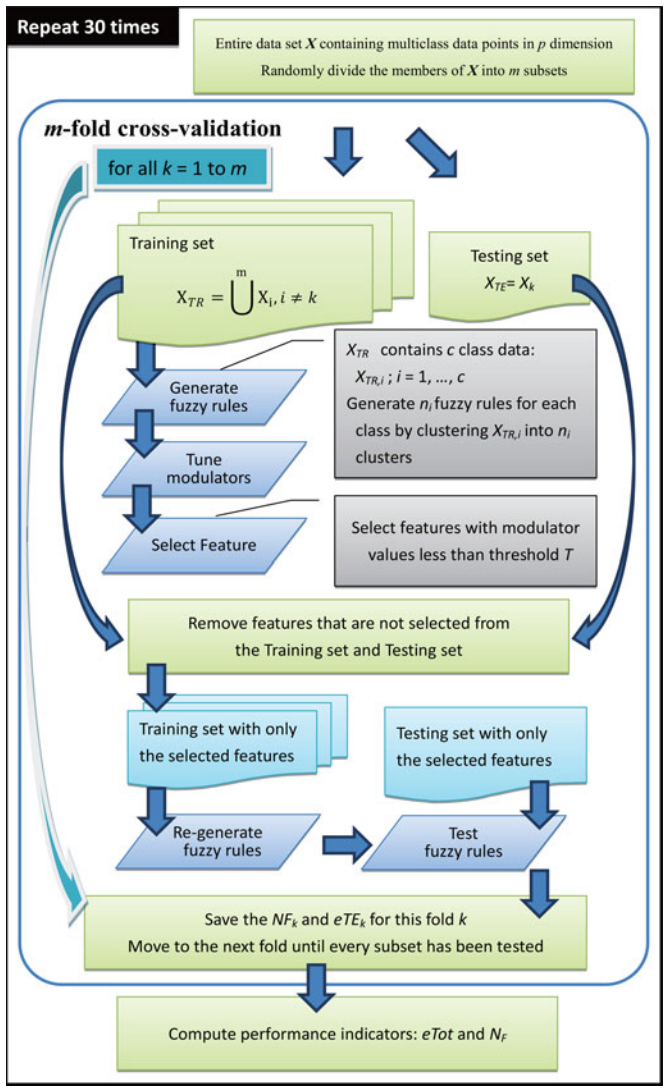


Fig. 4. Overall flow of the experiments.

Fig. 2 depicts the scatter plots of different pairs of features for the synthetic data. From these figures, it is clear that features F1 and F2 are the useful features, while others are not. For the synthetic data with four different thresholds, always two features, as expected, are selected and the selected features practically resulted in zero misclassification. In Fig. 5(a), we depict the frequencies with which different features are selected. This figure reveals that features F1 and F2 are selected always. Fig. 6(a) displays the opening of the modulators with iterations. Fig. 6(a) shows that within a few iterations the gates for the bad features (F3 and F4) are closed more tightly (value of λ becomes larger than 3), while the gates for the good features open up very fast. The modulators for features F1 and F2 take a value of practically zero within 100 iterations. We now list eight rules that are extracted by our system. Note that, for this dataset (as well as for the class Setosa of iris), only one rule per class would have been enough. Since finding optimal number of rules is not our objective, to keep consistency with other datasets, we

TABLE II
PERFORMANCE EVALUATION OF THE PROPOSED METHOD ON DIFFERENT DATASETS

Data Set (No. of features)	Threshold for feature selection	Average (std. devn.) of number of selected features	(%) reduction in number of features	Average test misclassification (%) (std. devn.) using selected features	Average test misclassification (%) (std. devn.) using all features
Synthetic (4)	1.7	2.00 (0.00)	50.00	0.00 (0.00)	0.68 (0.55)
	1.5	2.00 (0.00)	50.00	0.00 (0.00)	0.57 (0.45)
	1.3	2.00 (0.00)	50.00	0.00 (0.00)	0.50 (0.53)
Iris (4)	1.7	3.45 (0.42)	13.83	6.24 (1.38)	6.22 (1.10)
	1.5	3.39 (0.45)	15.17	6.16 (1.79)	6.11 (0.70)
	1.3	2.92 (0.20)	27.08	6.33 (2.54)	6.53 (0.92)
WDBC (30)	1.7	2.21 (0.21)	92.63	6.46 (0.72)	5.73 (0.47)
	1.5	2.14 (0.19)	92.86	6.45 (0.68)	5.55 (0.49)
	1.3	1.97 (0.14)	93.42	6.68 (0.77)	5.65 (0.47)
Wine (13)	1.7	4.38 (0.40)	66.33	4.49 (0.93)	4.42 (1.21)
	1.5	3.94 (0.35)	69.69	5.21 (1.27)	3.88 (0.94)
	1.3	3.27 (0.21)	74.82	6.54 (1.83)	4.10 (0.81)
Heart (6)	1.7	3.94 (0.30)	34.39	31.36 (2.29)	29.10 (1.70)
	1.5	2.77 (0.23)	53.89	34.41 (2.25)	29.72 (1.75)
	1.3	1.64 (0.18)	72.61	36.88 (2.73)	29.30 (2.02)
Ionosphere (33)	1.7	4.33 (0.43)	86.88	15.60 (1.60)	13.13 (1.39)
	1.5	3.01 (0.23)	90.89	16.62 (2.00)	13.38 (1.24)
	1.3	2.44 (0.27)	92.62	19.95 (2.45)	13.19 (1.41)
Sonar (60)	1.7	5.85 (0.47)	90.25	27.10 (2.42)	24.68 (2.53)
	1.5	3.57 (0.32)	94.06	29.70 (2.69)	24.44 (2.23)
	1.3	1.88 (0.30)	96.87	35.74 (3.28)	24.74 (2.66)
Glass (9)	1.7	6.96 (0.40)	22.70	38.46 (3.15)	37.04 (3.42)
	1.5	6.79 (0.49)	24.52	39.44 (2.81)	35.90 (2.48)
	1.3	6.63 (0.36)	26.30	38.83 (3.42)	36.84 (2.83)

have used two rules for every class. We get negative feature values in the rules because every feature is normalized using Z-score.

R_1: If F1 is CLOSE TO 0.97 AND F2 is CLOSE TO 1.00, then class is Class1.
R_2: If F1 is CLOSE TO 1.04 AND F2 is CLOSE TO 1.02, then class is Class1.
R_3: If F1 is CLOSE TO -1.02 AND F2 is CLOSE TO 0.96, then class is Class2.
R_4: If F1 is CLOSE TO -0.93 AND F2 is CLOSE TO 0.98, then class is Class2.
R_5: If F1 is CLOSE TO 1.01 AND F2 is CLOSE TO -1.04, then class is Class3.
R_6: If F1 is CLOSE TO 0.98 AND F2 is CLOSE TO -0.92, then class is Class3.
R_7: If F1 is CLOSE TO -1.00 AND F2 is CLOSE TO -0.97, then class is Class4.
R_8: If F1 is CLOSE TO -1.01 AND F2 is CLOSE TO -1.04, then class is Class4.

For the iris data, almost in every trial, features F3 and F4 are selected. However, Fig. 5(b) shows that features F1 and F2 are also selected in many trials. The conventional belief for the iris data is that features F3 and F4 are good and adequate. This is indeed the case and that is also revealed by our experiments. Fig. 6(b) depicts the opening of the gates with iteration in a typical run using the iris dataset. This figure shows that gates for features F3 and F4 open very fast and in about 200 and 100 iterations, respectively, these two gates are completely opened. At this point, the training square error as well as the misclassification reaches a very low value and further training does not improve the systems performance in terms of misclassification, although the training error decreases very slowly over the next 400 iterations and after that it does not change with iteration. This clearly suggests that F3 and F4 are adequate for this problem. Then why does our method select F1 and F2 in many trials? First, for the iris data, each of the four features has some discriminating power, and hence, at the beginning, all four gates start opening and the λs for F3 and F4 take a value of practically zero (0) in 200 and 100 iterations, respectively. Second, we have used a threshold of 1.3 or higher for all datasets. Therefore, we select F1 and F2 in several trials. If we use a lower threshold, say T = 1.0 or even a smaller threshold, our system usually selects only features F3 and F4, which is consistent with results mostly reported in the literature.

Fig. 6(b) also reveals an interesting point—the interactions between features and that between features and the learning framework. After feature F4 becomes completely active, further training (about 100 iterations) partially closes the gate associated with F2 making it less useful as it is not needed. We have included in the following a set of six rules that our system has extracted for the iris data. For this dataset, with $T = 1.3$, in some of the runs, features F3 and F4 are selected, and the rules correspond to one such run.

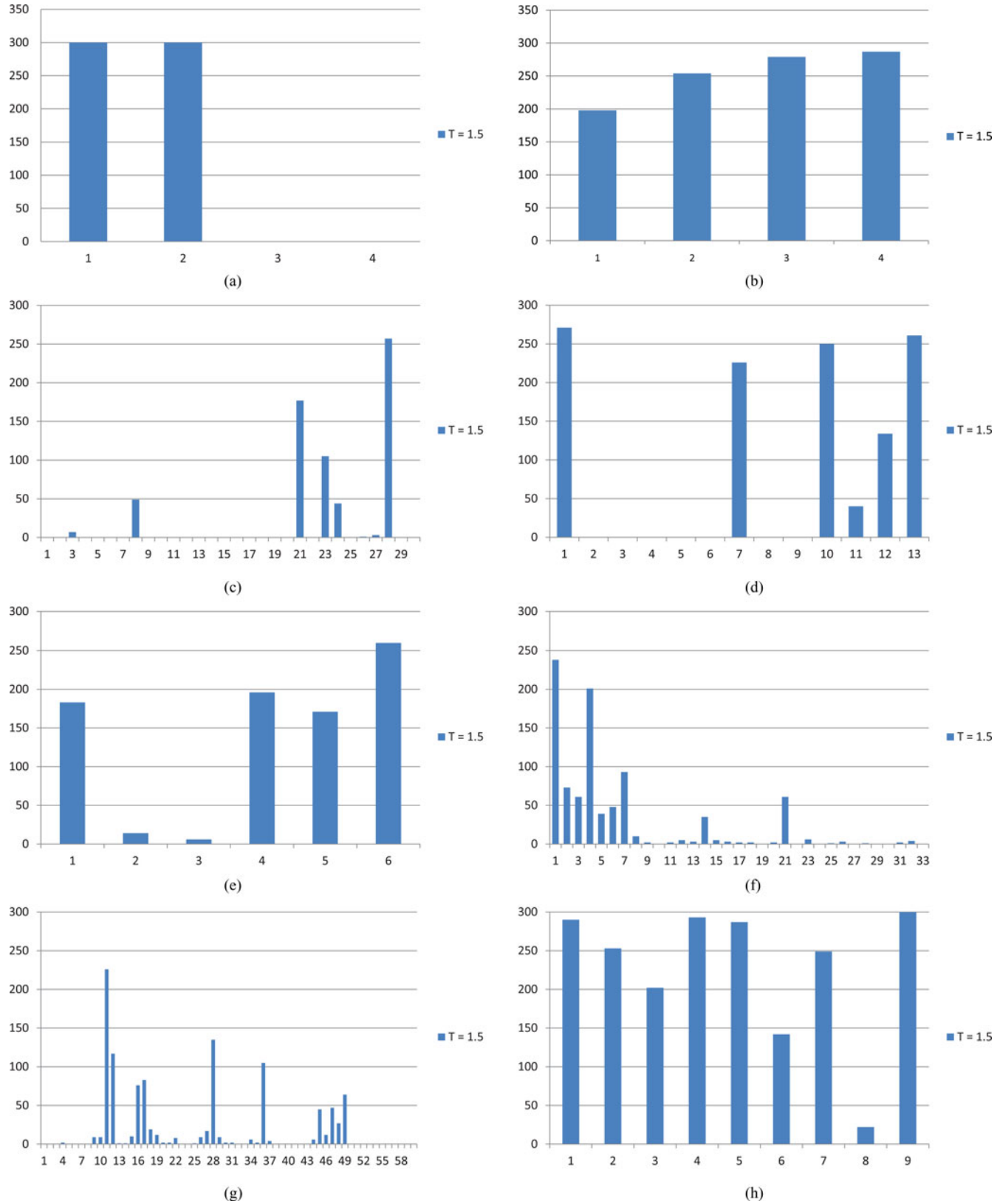


Fig. 5. Frequencies with which different features are selected in 300 trials for the feature selection experiments. (a) For the synthetic data. (b) For the iris data. (c) For the WDBC data. (d) For the wine data. (e) For the heart data. (f) For the ionosphere data. (g) For the sonar data. (h) For the glass data.

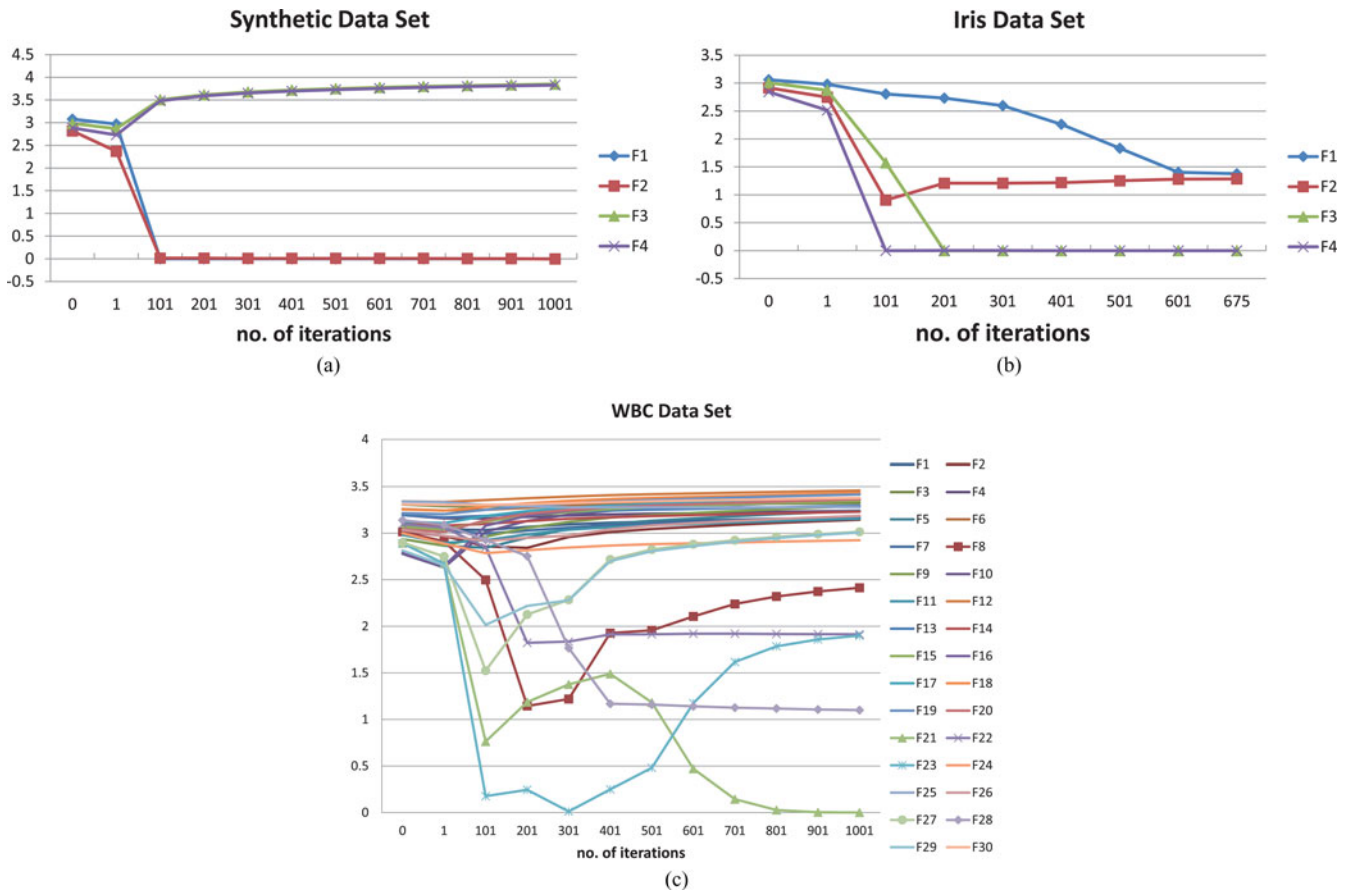


Fig. 6. Opening of the feature modulating gates in a typical run using all data points. (a) For the synthetic data. (b) For the iris data. (c) For the WDBC data.

TABLE III
EFFECT OF σ ON THE PERFORMANCE OF THE PROPOSED METHOD FOR THE SONAR DATA

Data Set (No. of features)	σ^2	Average (std. devn.) of number of selected features	(%) reduction in number of features	Average test misclassification (%) (std. devn.) using selected features	Average test misclassification (%) (std. devn.) using all features
Sonar (60)	1.0	3.57 (0.32)	94.06	29.70 (2.69)	24.44 (2.23)
	2.0	4.08 (0.44)	93.21	28.19 (3.39)	24.70 (2.65)
	3.0	5.75 (0.41)	90.42	26.68 (2.74)	24.46 (2.07)
	4.0	6.99 (0.35)	88.35	26.83 (2.83)	24.39 (2.53)
	5.0	8.04 (0.46)	86.61	26.41 (2.51)	24.62 (2.02)

R_1: If F3 is CLOSE TO -1.31 AND F4 is CLOSE TO -1.31, then class is Setosa.
R_2: If F3 is CLOSE TO -1.30 AND F4 is CLOSE TO -1.19, then class is Setosa.
R_3: If F3 is CLOSE TO 0.40 AND F4 is CLOSE TO 0.29, then class is Versicolor.
R_4: If F3 is CLOSE TO 0.10 AND F4 is CLOSE TO -0.01, then class is Versicolor.
R_5: If F3 is CLOSE TO 1.17 AND F4 is CLOSE TO 1.25, then class is Virginica.
R_6: If F3 is CLOSE TO 0.79 AND F4 is CLOSE TO 0.85, then class is Virginica.

In this study, our objective is to focus only on feature selection in an integrated manner with rule extraction. Hence, we have ignored all other issues relating to rule extraction such as tuning membership parameters and minimization of the number of rules for each class.

For the WDBC dataset, on average with just two to three features (a reduction of about 93% of features), we get an average test accuracy of 6.4% $\sim$ 6.7%, while using all features the average test accuracy is 5.6%—it is practically the same. Table II shows the average number of features selected over the 30 repetition of the ten-fold cross-validation experiments; i.e., over 300 (= 30 $\times$ 10) trials of feature selection. Unless there are too many correlated and useful features, we expect a small subset of features to be selected in most runs. Fig. 5(c) reveals that feature F28 is selected in most of the 300 experiments, while F21 and F23 are selected almost 50% times and 33% times, respectively. For this dataset, Fig. 6(c) depicts the

opening of modulating gates with iterations for a typical run. From this figure, we find that gates for features F21 and F23 open very fast at the beginning and in about 100 iterations the gate for F23 becomes almost completely opened indicating that it is probably the most important discriminator. However, after this, F21 and F23 start interacting and after 300 iterations, the gate for F23 starts closing, while that for F21 starts opening further. In about 800 iterations, F21 becomes completely active, while F23 becomes inactive! Similar phenomena can be observed for features F8 and F27. What does it mean? It suggests that initially all of F8, F21, F23, and F27 are important, but when some other features become active (for example, F28), the relevance of F8, F23, and F27 becomes less. In fact, use of F21 in conjunction with F8, F22, F23, and F27 might have become counterproductive as far as reduction of training error using this rule-based framework. This illustrates how such an integrated mechanism can take into account the subtle interaction among modeling tool (here fuzzy rules), features and the problem being solved. In many runs for this dataset, features F21 and F28 are selected. The rules in the following correspond to one such case. These rules suggest that these two features can separate the benign and malignant classes effectively.

R_1: If F21 is CLOSE TO -0.75 AND F28 is CLOSE TO -0.14, then class is Benign.
R_2: If F21 is CLOSE TO -0.58 AND F28 is CLOSE TO -0.68, then class is Benign.
R_3: If F21 is CLOSE TO 1.76 AND F28 is CLOSE TO 1.34, then class is Malignant.
R_4: If F21 is CLOSE TO 0.44 AND F28 is CLOSE TO 0.80, then class is Malignant.

For the wine data, our method selects on average 3.27 and 3.94 features when the threshold used on the modulator is 1.3 and 1.5, respectively. The average misclassification is about 5.2–6.5%. It may be noted that, using all features, the average test error is also about 4.1% on average. Fig. 5(d) depicts the frequency with which different features are selected for the wine dataset when $T = 1.5$ is used. From this figure, we find that four features, i.e., F1, F7, F10, and F13, are selected in most of the 300 experiments suggesting that these features are probably necessary features. Although F11 and F12 are selected about one sixth and one half of the times, respectively, the remaining features are practically never selected.

In case of the heart data, Fig. 5(e) shows that features F2 and F3 are usually rejected by the system, while feature F6 is selected in most of the 300 trials. As shown in Table II, for $T = 1.7$ with four features, the test performance of the rule base is very similar to that with all features.

For the ionosphere data, for $T = 1.7$ on average using about 4.3 features, the FRBS is able to get 15.6% test error, while using all 33 features, the rule-based system can achieve about 13.1% test error. Fig. 5(f) shows the frequency with which different features are selected over the 300 experiments. From Fig. 5(f), we find that many of the features, such as F9, F10, F11, F17, F18, F19, F20, F22, F24, F25, F27, F28, F29, F30, F31, and F33, are practically not selected. These features seem derogatory features.

For thesonar data, use of $T = 1.5$ on average selects only 3.6 features (about 94% reduction of features), yielding an average test error of 29.7%, while using all 60 features the average test error is about 24.5%. Since there are 60 features, we relaxed the threshold to $T = 1.7$ and with this choice, on average, it selects about six features (a reduction of 90%) and, thereby, reducing the test error to about 27.1%, which is quite close to the performance with all features (24.7%). For this dataset, from Fig. 5(g), we find that many features are rejected by the system for $T = 1.5$. In particular, features F1–F3, F5–F8, F23, F24, F32, F33, F38–F43, F50–F60, and several others are practically rejected by our fuzzy rule-based scheme. To see the effect of the choice of fixed but different spreads for the membership functions, we have performed some additional experiments for the sonar dataset using different σ ($\sigma^2 = 1 \sim 5$). These results are shown in Table III, which reveal that for a fixed $T = 1.5$, with larger σ, the system tends to select a few more features.

For the glass data (see Table II), on average, about seven features are selected. Fig. 5(h) shows that feature F8 is rejected by our system and features F3 and F6 are selected about two thirds and one half of the time, respectively. The remaining six features are selected most of the times. Fig. 3 shows a scatter plot of the top two principal components of the normalized glass data, where different classes are represented by different shapes and colors. This picture clearly reveals the substantial overlap between different classes—it is not an easy task to discriminate between the classes.

Use of less number of features reduces the complexity of the system and enhances the comprehensibility of the systems, but *usually*, it also reduces the accuracy. That is why we often need to make a tradeoff between complexity and accuracy [28]–[35]. The reason we say "usually" is that elimination of features does not necessarily reduce the accuracy. For example, removal of derogatory features or indifferent features can only enhance the accuracy. Elimination of such features cannot reduce the accuracy. In Table II, the results for the synthetic data reveal that feature elimination can improve the accuracy. Similarly, for the wine data, with about 66% reduction in the number of features, practically, there is no decrease in accuracy. However, for other datasets, we find that the use of less features decreases the accuracy. In other words, enhanced interpretability (reduced complexity) may result in a decrease in accuracy when we remove features with good or moderate discriminating power. When we try to achieve complexity minimization and accuracy maximization through feature selection, usually (except elimination of derogatory and indifferent features), these two are conflicting objectives. As mentioned earlier, there are many multiobjective approaches to design fuzzy systems addressing the accuracy–complexity tradeoff in the literature [28]–[35]. Such approaches, simultaneously, consider the number of correctly classified training patterns, the number of fuzzy rules, even the total number of antecedent conditions (and/or the number of features), while designing the system. In this study, we do not address this issue. Here, we want to discard derogatory (or bad) features and indifferent features and select a set of features that can solve the problem at hand satisfactorily. Since we focus strictly on this, we have ignored other issues such as

minimization of the number of rules and tuning of membership functions and, hence, the accuracy–complexity tradeoff.

V. Conclusion and Discussion

We have proposed an integrated scheme for simultaneous feature selection and fuzzy rule extraction. We have demonstrated that unlike other feature selection methods used in connection with fuzzy rules, the proposed scheme can capture the subtle interaction between features as well as the interaction between features and the modeling tool (here a fuzzy rule base) and, hence, can find a small set of useful features for the task at hand. The effectiveness of the method is demonstrated using several datasets from the UCI machine learning repository as well as using a synthetic dataset. Our rule generation scheme is not a partition-based scheme. We cluster the data from a class into a number of clusters and each cluster is converted into a rule. Thus, the increase in the dimensionality of the feature space does not have a direct impact on the number of rules. Hence, the rule generation is scalable. In case of evolutionary algorithm-based methods, the search space increases exponentially with the number of features, and hence, there is a substantial increase in computation time with dimension. Similarly, the time complexity of wrapper methods, which evaluate various subsets of features, increases rapidly with dimensionality of the data. However, our method uses a single-pass gradient-based search technique and, hence, is not affected much by the dimensionality of the data. Thus, the proposed approach is a scalable method both from the point of view of rule generation and feature selection.

Note that the objective of this study is not to find an "optimal" rule base for the classification problem but to find a set of useful features that can solve the problem at hand satisfactorily. That is why we did not tune parameters (center and spread of membership functions) of the rule base. A set of useful features must not have any bad (derogatory) feature; it must also not include any indifferent features. However, it may have some redundant features. We also note that, from implementation point of view, the set of selected (useful) features is dependent on the choice of the threshold unless we are dealing with datasets like the synthetic one, which has only good and bad features but no redundant feature.

In this investigation, we have used datasets up to dimension 60, but what will happen if the dimension is very high, say a few thousand! One possibility may be to divide the features arbitrarily into a number of subsets each of size say 100. The scheme may be applied on each such subset to select a small number of features from each such subset. Then, again, the scheme can be applied on the set of selected features. Such a hierarchical approach may not be optimal but is likely to yield a useful solution. This is left for our future investigation.

We began our discussion with a classification of features. Now, let us see what kind of features our scheme could select. First, it must select the necessary features unless the optimization lands up in a very poor minima. If and when this happens, the problem cannot be solved using the selected features. Such a situation can be detected by looking at the training error, which will be very high. Such runs can thus be discarded easily. We emphasize that we have never landed up on such a poor minima in all of the experiments that we have conducted. Second, our system will not select bad features. Why? Because we start assuming every feature as bad/poor. Since such a poor feature cannot reduce the training error, the system is not likely to open any such gate (modulator). Third, our system will not select indifferent features because when we start our training, we set indifferent features, if any, as bad features. Since such features cannot reduce the training error, they are also not likely to get selected. However, our system may select some redundant features. Suppose there are four features x, y, z, and s. Suppose the feature y has a high correlation to feature z and both are good features. The feature x is an essential feature and s is a bad feature. To solve the problem, we need either x, y or x, z. In this case, if we run our method, depending on the initial condition, it may select $\{x, y\}$, $\{x, z\}$ as well as $\{x, y, z\}$. Note that each of the three sets are equally good to solve the problem, but which of the three is the most desirable? Well, if we do not like to have redundancy in our system (which may not necessarily be good), we want to select either $\{x, y\}$ or $\{x, z\}$. In a more general setting, we can say that we should select features with some (controlled) redundancy as that may help to deal with measurement errors for some features. Selection of features with controlled redundancy is going to be the theme of a future paper.

References

[1] N. R. Pal and K. K. Chintalapudi, "A connectionist system for feature selection," *Neural, Parallel Sci. Comput.*, vol. 5, pp. 359–382, 1997.

[2] 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.

[3] 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.

[4] N. R. Pal, E. V. Kumar, and G. Mandal, "Fuzzy logic approaches to structure preserving dimensionality reduction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 3, pp. 277–286, Jun. 2002.

[5] N. R. Pal and E. V. Kumar, "Two efficient connectionist schemes for structure preserving dimensionality reduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1142–1153, Nov. 1998.

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

[7] D. Chakraborty and N. R. Pal, "Selecting useful groups of features in a connectionist framework," *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 381–396, Mar. 2008.

[8] D. Muni, N. R. Pal, and J. Das, "Genetic programming for simultaneous feature selection and classifier design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 106–117, Feb. 2006.

[9] I. Guyon, J. Weston, S. Barnhill, and V. Vapnil, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, pp. 389–422, 2002.

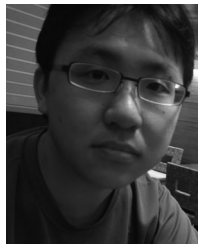
[10] S. Niijima and S. Kuhara, "Recursive gene selection based on maximum margin criterion: A comparison with SVM-RFE," *BMC Bioinformat.*, vol. 7, 543, 2006.

[11] N. R. Pal and S. Saha, "Simultaneous structure identification and fuzzy rule generation for Takagi–Sugeno models," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 6, pp. 1626–1638, Dec. 2008.

[12] N. R. Pal, "A fuzzy rule based approach to identify biomarkers for diagnostic classification of cancers," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2007, pp. 1–6.

[13] G. V. S. Raju, J. Zhou, and R. A. Kisner, "Hierarchical fuzzy control," *Int. J. Control*, vol. 54, pp. 1201–1216, 1991.

[14] G. V. S. Raju and J. Zhou, "Adaptive hierarchical fuzzy controller," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 4, pp. 973–980, Jul./Aug. 1993.
[15] Y. Yoshinari, W. Pedrycz, and K. Hirota, "Construction of fuzzy models through clustering techniques," *Fuzzy Sets Syst.*, vol. 54, no. 2, pp. 157–166, 1993.
[16] R. R. Yager and D. P. Filev, "Generation of fuzzy rules by mountain clustering," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 209–219, 1994.
[17] R. Babuska and U. Kaymak, "Application of compatible cluster merging to fuzzy modeling of multivariable systems," in *Proc. Eur. Congr. Intell. Tech. Soft Comput.*, 1995, pp. 565–569.
[18] S. L. Chiu, "Extracting fuzzy rules for pattern classification by cluster estimation," in *Proc. Int. Fuzzy. Syst. Assoc.*, vol. 2, Sao Paulo, Brazil, 1995, pp. 273–276.
[19] S. L. Chiu, "Selecting input variables for fuzzy models," *J. Intell. Fuzzy Syst.*, vol. 4, no. 4, pp. 243–256, 1996.
[20] M. Delgado, A. F. Gomez-Skarmeta, and F. Martin, "A fuzzy clustering based rapid-prototyping for fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 2, pp. 223–233, May 1997.
[21] H. Pomares, I. Rojas, J. Ortega, J. Gonzalez, and A. Prieto, "A systematic approach to a self-generating fuzzy rule-table for function approximation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 3, pp. 431–447, Jun. 2000.
[22] M. G. Joo and J. S. Lee, "Hierarchical fuzzy logic scheme with constraints on the fuzzy rule," *Int. J. Intell. Autom. Soft Comput.*, vol. 7, no. 4, pp. 259–271, 2001.
[23] M. G. Joo and J. S. Lee, "Universal approximation by hierarchical fuzzy system with constraints on the fuzzy rule," *Fuzzy Sets Syst.*, vol. 130, no. 2, pp. 175–188, 2002.
[24] M. G. Joo and J. S. Lee, "Class of hierarchical fuzzy systems with constraints on the fuzzy rules," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 2, pp. 194–203, Apr. 2005.
[25] K. Pal, R. Mudi, and N. R. Pal, "A new scheme for fuzzy rule based system identification and its application to self-tuning fuzzy controllers," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 4, pp. 470–482, Aug. 2002.
[26] N. R. Pal, R. Mudi, K. Pal, and D. Patranabish, "Rule extraction to exploratory data analysis for self-tuning fuzzy controllers," *Int. J. Fuzzy Syst.*, vol. 6, no. 2, pp. 71–80, 2004.
[27] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 260–270, Aug. 1995.
[28] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 5, pp. 601–618, Oct. 1999.
[29] H. Ishibuchi, T. Yamamoto, and T. Nakashima, "Hybridization of fuzzy GBML approaches for pattern classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 359–365, Apr. 2005.
[30] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets Syst.*, vol. 89, no. 2, pp. 135–150, 1997.
[31] H. Ishibuchi, T. Nakashima, and T. Murata, "Three-objective genetics-based machine learning for linguistic rule extraction," *Inf. Sci.*, vol. 136, no. 1–4, pp. 109–133, 2001.
[32] H. Ishibuchi and T. Yamamoto, "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining," *Fuzzy Sets Syst.*, vol. 141, no. 1, pp. 59–88, 2004.
[33] H. Ishibuchi and Y. Nojima, "Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning," *Int. J. Approx. Reason.*, vol. 44, no. 1, pp. 4–31, 2007.
[34] E. G. Mansoori, M. J. Zolghadri, and S. D. Katebi, "SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 4, pp. 1061–1071, Aug. 2008.
[35] M. J. Gacto, R. Alcala, and F. Herrera, "Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, pp. 515–531, Jun. 2010.
[36] R. Alcala, M. J. Gacto, and F. Herrera, "A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 4, pp. 666–681, Aug. 2011.
[37] J. Alcala-Fdez, R. Alcala, and F. Herrera, "A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 5, pp. 857–872, Oct. 2011.
[38] Y. Jin, "Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 2, pp. 212–221, Apr. 2000.
[39] S. M. Vieira, J. M. C. Sousa, and T. A. Runkler, "Ant colony optimization applied to feature selection in fuzzy classifiers," in *Proc. 12th Int. Fuzzy. Syst. Assoc.*, 2007, vol. 4259, pp. 778–788.
[40] 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.
[41] E. Lughofer and S. Kindermann, "SparseFIS: Data-driven learning of fuzzy systems with sparsity constraints," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 396–411, Apr. 2010.
[42] G. D. Wu and P. H. Huang, "A maximizing-discriminability-based self-organizing fuzzy network for classification problems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 362–373, Apr. 2010.
[43] H. Han and J. Qiao, "A self-organizing fuzzy neural network based on a growing-and-pruning algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 6, pp. 1129–1143, Dec. 2010.
[44] H. Pomares, I. Rojas, J. Gonzalez, and A. Prieto, "Structure identification in complete rule-based fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 3, pp. 349–359, Jun. 2002.
[45] H. M. Lee, C. M. Chen, J. M. Chen, and Y. L. Jou, "An efficient fuzzy classifier with feature selection based on fuzzy entropy," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 3, pp. 426–432, Jun. 2001.
[46] L. Sanchez, M. R. Suaez, J. R. Villar, and I. Couso, "Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data," *Int. J. Approx. Reason.*, vol. 49, no. 3, pp. 607–622, 2008.
[47] R. Silipo and M. R. Berthold, "Input features' impact on fuzzy decision processes," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 6, pp. 821–834, Dec. 2000.
[48] Q. Hu., Z. Xie and D. Yu, "Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation," *Pattern Recognit.*, vol. 40, no. 12, pp. 3509–3521, 2007.
[49] R. Jensen and Q. Shen, "Fuzzy-rough sets assisted attribute selection," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 1, pp. 73–89, Feb. 2007.
[50] R. Jensen and Q. Shen, "New approaches to fuzzy-rough feature selection," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 4, pp. 824–838, Aug. 2009.
[51] N. R. Pal, K. Pal, J. C. Bezdek, and T. Runkler, "Some issues in system identification using clustering," in *Proc. Int. Joint Conf. Neural Netw.*, 1997, pp. 2524–2529.
[52] N. R. Pal and S. Chakraborty, "Fuzzy rule extraction from ID3 type decision trees for real data," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 31, no. 5, pp. 745–754, Oct. 2001.
[53] Z. Deng, F. L. Chung, and S. Wang, "Robust relief-feature weighting, margin maximization, and fuzzy optimization," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 726–744, Aug. 2010.
[54] L. Kuncheva, *Fuzzy Classifier Design*. New York: Physica-Verlag, 2000.
[55] 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.
[56] A. Ghosh, N. R. Pal, and J. Das, "A fuzzy rule based approach to cloud cover estimation," *Remote Sens. Environ.*, vol. 100, no. 4, pp. 531–549, 2006.
[57] (2011). [Online]. Available: http://archive.ics.uci.edu/ml/



Yi-Cheng Chen received the B.S. degree from the Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan, and the M.S. degree from the Institute of Biomedical Informatics, National Yang-Ming University, Taipei, in 2006 and 2008, respectively. He is currently working toward the Ph.D. degree with the Institute of Biomedical Informatics, National Yang-Ming University.

His current research interests include information technology, computational intelligence, and bioinformatics.

Nikhil R. Pal (F'05) is currently a Professor with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta, India. He has coauthored/edited or coedited several books. He has given plenary and keynote speeches at many international conferences. His current research interests include bioinformatics, brain science, pattern recognition, fuzzy sets theory, neural networks, and evolutionary computation.

Prof. Pal serves the editorial/advisory board of several journals, including the *International Journal of Approximate Reasoning*, the *International Journal of Hybrid Intelligent Systems*, the *International Journal of Neural Systems*, and the *International Journal of Fuzzy Sets and Systems*. He is a fellow of the Indian National Science Academy, the Indian National Academy of Engineering, and the International Fuzzy Systems Association. He is an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS. He served as the Editor-in-chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS for six years (2005–2010). He is an IEEE Computational Intelligence Society (CIS) Distinguished Lecturer (2010–1012) and an elected member of the IEEE-CIS Administrative Committee (2010–2012).

I-Fang Chung (M'10) received the B.S., M.S., and Ph.D. degrees in control engineering from the National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1993, 1995, and 2000, respectively.

From 2000 to 2003, he was a Research Assistant Professor with the Department of Electrical and Control Engineering, NCTU. From 2003 to 2004, he was a Postdoctoral Fellow with the Laboratory of DNA Information Analysis, Human Genome Center, Institute of Medical Science, Tokyo University, Tokyo, Japan. In 2004, he joined the Institute of Biomedical Informatics, National Yang-Ming University, Taipei, Taiwan, where, since 2011, he has been an Associate Professor. He is also with the Center for Systems and Synthetic Biology, National Yang-Ming University. His current research interests include computational intelligence, bioinformatics, biomedical engineering, and biomedical signal processing.