MODULAR FEATURE SELECTION USING RELATIVE IMPORTANCE FACTORS

Sheng-Uei Guan[1], Fangming Zhu, Peng Li
Department of Electrical and Computer Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 119260

Feature selection plays an important role in finding relevant or irrelevant features in classification. Genetic algorithms (GAs) have been used as conventional methods for classifiers to adaptively evolve solutions for classification problems. In this paper, we explore the use of feature selection in modular GA-based classification. We propose a new feature selection technique, Relative Importance Factor (RIF), to find irrelevant features in the feature space of each module. By removing these features, we aim to improve classification accuracy and reduce the dimensionality of classification problems. Benchmark classification data sets are used to evaluate the proposed approaches. The experiment results show that RIF can be used to determine irrelevant features and help achieve higher classification accuracy with the feature space dimension reduced. The complexity of the resulting rule sets is also reduced which means the modular classifiers with irrelevant features removed will be able to classify data with a higher throughput.

Keywords: classification, feature selection, genetic algorithm, class decomposition

1. Introduction

Classification problems play a major role in various fields of computer science and engineering, such as image processing and data mining. A number of soft computing approaches, such as neural networks (Anand *et al.*, 1995; Lu and Ito, 1999; Guan and Li, 2002a), evolutionary algorithms (Corcoran and Sen, 1994), and fuzzy logic (Ishibuchi *et al.*, 1999; Setnes and Roubos, 2000), have been widely used to adaptively evolve solutions for classification problems. Among them, GA-based solutions have attracted much attention and become one of the popular techniques for classification (Merelo *et al.*, 2001).

However, when GA is applied to larger-scale real-world classification problems, it still suffers from some drawbacks, such as the inefficiency in searching a large space, the difficulty in breaking internal interference of training data, and the possibility of getting trapped in local optima. A natural approach to overcome these drawbacks is to decompose the original task into several sub-tasks based on certain techniques. Generally, a decomposition approach divides a task into smaller and simpler sub-tasks, supervises the learning of each sub-task, and finally recombines individual solutions into the final solution. Various task decomposition methods have been proposed. These methods can be roughly classified into the following categories: functional modularity, domain modularity, class decomposition, and state decomposition, according to different partition strategies (Anand *et al.*, 1995; Guan and Li, 2002a; Jenkins and Yuhas, 1993; Lu and Ito, 1999).

A number of features are usually available for classification problems. However, not all of the features are equally important for a specific task. Some of them may be redundant or even irrelevant. Better performance may be achieved by discarding some features (Verikas and Bacauskiene, 2002). In other circumstances, we may aim to reduce the dimensionality of input space to save some computation effort, although classification accuracy may be slightly deteriorated. There are many feature selection techniques developed from various perspectives such as performance (Setiono and Liu, 1997), mutual information (entropy) (Battiti, 1994; Kwak and Choi, 2002), and statistic information (Lerner *et al.*, 1994).

Principal component analysis (PCA) and linear discriminant analysis are two traditional techniques used to reduce dimensionality by creating new features that are linear combinations of the original ones (Fukunaga, 1990). Fisher's linear discriminant (FLD) is the most popular goodness-score function used

[1] Corresponding author: eleguans@nus.edu.sg

in feature selection. It is simple in computation and does not need strict assumptions in the distribution of features. Generally, various combinations of features in the original feature space can be evaluated with the goodness-score function by excluding some features in the feature space. Because all possible combinations of the features should be tried, the computation effort of such techniques is very high. In order to reduce computation time, some search algorithms are developed, such as knock-out and backtrack tree (Lerner *et al.*, 1994; Gonzalez and Perez, 2001).

Some feature selection techniques based on neural network and fuzzy set theory have been proposed. Setiono and Liu (1997) proposed a technique based on the performance evaluation of a neural network. In their technique, the original features are excluded one by one and the neural network is retrained and evaluated repeatedly. Pal et al. (2000) demonstrated a way of formulating neuro-fuzzy approaches for feature selection under unsupervised learning. A fuzzy feature evaluation index for a set of features is defined in terms of degree of similarity between two patterns.

In this paper, we employ a modular GA-based scheme for classification. This modular scheme uses class decomposition, which partitions a classification problem into several class modules in the output domain. Each module is responsible for solving a fraction of the original problem. These modules can be trained in parallel and independently, and the results obtained from them are integrated to form the final solution. Then, we propose a new feature selection technique - Relative Importance Factor (RIF) based on the optimal transformation weights from Fisher's linear discriminant function. The RIF technique can detect features that are irrelevant to the classification problem and remove them from the feature space to improve classification performance in terms of accuracy and complexity. We integrate RIF into the modular GA-based scheme by employing it in finding a suitable feature subset for each class module. We aim to explore the application of feature selection in the GA domain, which appears to be missing in the literature. A modular-GA based classification approach will be more effective for RIF feature selection, as it is easier to find the irrelevant features (IRFs) in individual class, eliminating the interference from other classes. Three benchmark data sets are used to evaluate the performance of RIF. The experiment results show that RIF can help achieve higher classification accuracy with the feature space dimension reduced.

We first introduce the genetic approach for rule-based classification in section 2, and class decomposition for GA-based classification is elaborated in section 3. Then, a new feature selection technique RIF is introduced in section 4. The experiment results on benchmark data sets and their analysis are reported in section 5. Section 6 concludes the paper and presents future work.

2. A Genetic Approach to Rule-based Classification

2.1 Encoding Mechanism

In rule-based classification, there are various representation methods in terms of rule properties (fuzzy or non-fuzzy) and feature properties (nominal or continuous). In our approach, we use non-fuzzy IF-THEN rules with continuous features. A rule set consisting of a certain number of rules is a solution candidate for a classification problem. An IF-THEN rule is represented as follows:

$$R_i: \text{IF } (V_{1\min} \le x_1 \le V_{1\max}) \wedge (V_{2\min} \le x_2 \le V_{2\max}) \wedge \ldots \wedge (V_{n\min} \le x_n \le V_{n\max}) \text{ THEN } y = C \qquad (1)$$

where R_i is a rule label, n is the number of features, (x_1, x_2,... x_n) is the input feature set, and y is the output class category assigned with a value of C. V_{jmin} and V_{jmax} are the minimum and maximum bounds of the jth feature x_j respectively. We encode rule R_i according to the following diagram:

Antecedent Gene 1			……	Antecedent Gene n			Consequence Gene
Act_1	V_{1min}	V_{1max}	……	Act_n	V_{nmin}	V_{nmax}	C

Notes: 1. Act_j denotes whether condition j is active or inactive, which is encoded as 1 or 0.

2. If V_{jmin} is larger than V_{jmax} at any time, this gene will be regarded as an invalid gene. Invalid genes will make no contribution in a classification rule.

Each antecedent gene represents a feature, and the consequence gene stands for a class. Each chromosome CR_j consists of a set of classification rules R_i *(i=1,2…,m)* by concatenation:

$$CR_j = \bigcup_{i=1,m} R_i \qquad j = 1,2,\ldots,s \qquad (2)$$

where m is the maximum number of rules allowed for each chromosome, s is the size of the population. Therefore, one chromosome will represent one rule set. Since we know the discrete value range for

each feature and class *a priori*, V_{jmin}, V_{jmax}, and C can be encoded each as a character by finding their positions in the ranges. Thus, the final chromosome can be encoded as a string.

2.2 Genetic Operators

We use one-point crossover in all experiments. Referring to the encoding mechanism, we note that crossover will not cause inconsistency and thus can take place in any point of chromosome. The mutation point is randomly selected with a certain probability. According to the position of a selected point, we can determine whether it is an activeness, minimum, or maximum element. Different mutation is available for each. For example, if an activeness element is selected for mutation, it will just be toggled. Otherwise when a boundary-value element is selected, the algorithm will randomly select a substitute in the range of that feature. Figure 1 shows the operations of crossover and mutation. The rates for mutation and crossover are selected as 0.01 and 1.0 in our experiments. For reproduction, we set the survival rate as 50% (SurvivorsPercent=50%), which means half of the parent chromosomes with higher fitness will survive into the new generation, while the other half will be replaced by the newly created children resulting from crossover and/or mutation.

Roulette wheel selection (Michalewicz, 1996) is used in this paper. In this investigation, the probability that a chromosome will be selected for mating is given by the chromosome's fitness divided by the total fitness of all the chromosomes. By this means, chromosomes with higher fitness have a higher probability of producing offspring during selection for the next generation than those with lower fitness.

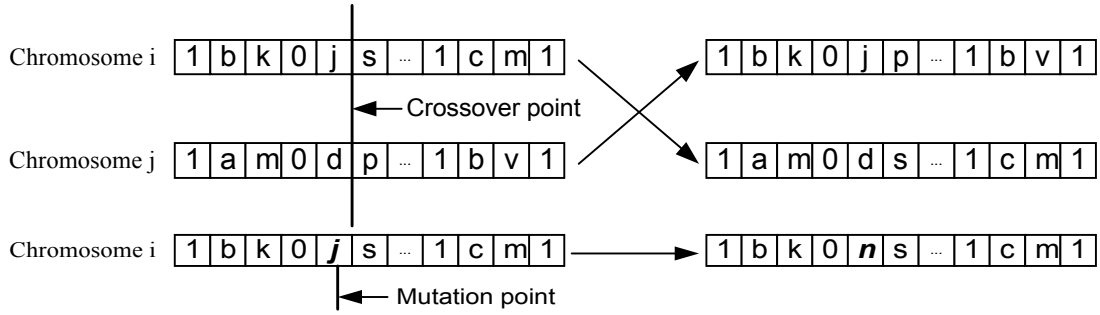

Figure 1. Crossover and mutation

2.3 Fitness Function

As each chromosome in our approach comprises an entire rule set, the fitness function actually measures the collective behavior of the rule set. The fitness function simply measures the percentage of instances that can be correctly classified by the chromosome's rule set, which can be represented as:

$$f = \frac{C}{N} = \frac{number\ \ of\ \ instances\ \ correctly\ \ classified}{total\ \ number\ \ of\ \ instances} \qquad (3)$$

Since there is more than one rule in a chromosome, it is possible that multiple rules match the conditions for all features but predicting different classes. We use a voting mechanism to resolve conflict. That is, each rule casts a vote for the class predicted by itself, and finally the class with the highest votes is regarded as the conclusive class. If there is a tie on one instance, it means that this instance cannot be classified correctly by this rule set.

2.4 Stopping Criteria

There are three factors in the stopping criteria. The evolution process stops after a preset generation limit, or when the best chromosome's fitness reaches a preset threshold (which is set as 1.0 through this paper), or when the best chromosome's fitness has no improvement over a specified number of generations -- stagnation limit. The detailed settings are reported along with corresponding experimental results.

3. Modular GA-based Classification with Class Decomposition

Let us assume a classification problem has c classes in the n-dimensional feature space. And p vectors $X_i = (x_{i1},\ x_{i2},\ ...,\ x_{in})$, $i = 1,2,...,\ p$, $p >> c$, are given as training patterns. The task of classification is to assign instances to one out of the pre-defined c classes, by discovering certain relationship among the features. Then, the discovered rules can be evaluated by classification accuracy or error rate either on the training data or test data.

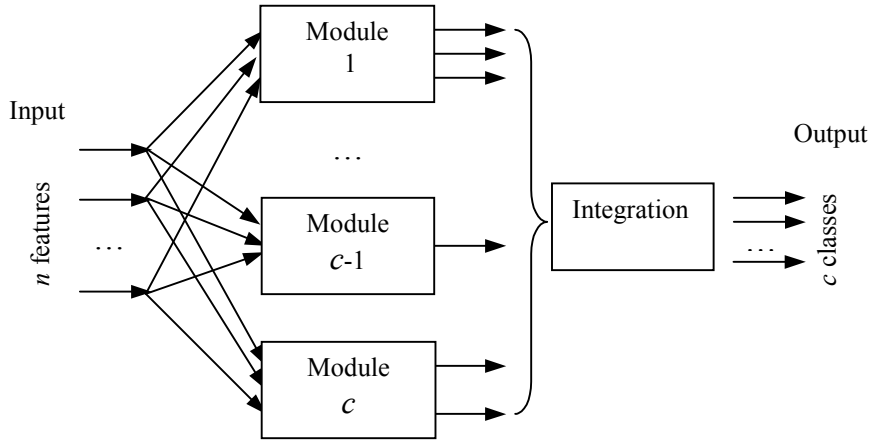

Figure 2. Illustration of modular GA-based classification with class decomposition

A traditional GA maps attributes to classes directly in a batch manner, which means all the attributes, classes, and training data are used together to train a group of GA chromosomes. Our approach -- GA with class decomposition is significantly different. As shown in Figure 2, it generally consists of three steps. Firstly, the original problem is divided into *c* sub-problems in terms of classes. Then, *c* GA modules are constructed for these sub-problems, and GA in each module will be responsible for evolving a sub-solution. Finally, these sub-solutions are integrated to form the final solution for the original problem. We present the details for each step in the following subsections.

3.1 Class Decomposition

The first step is to decompose a classification problem with a high-dimensional class space into sub-problems with low-dimensional class spaces, in terms of class categories.

Following the notations presented above, the original classification problem can be denoted as:

$$f: \quad X \rightarrow T \tag{4}$$

where, $X \in R^n$ is the set of features, and $T \in R^c$ is the set of classes. The objective of GA is to find a certain *f* with a satisfactory classification rate on the whole training set ξ, which can be represented as:

$$\xi = \{(X_i, T_i)\}_{i=1}^{p} \tag{5}$$

Now the *c*-class problem is fully decomposed into *c* sub-problems. Denoting the class for each sub-problem as $T^{(j)}$, we have:

$$T = T^{(1)} \cup T^{(2)} \cup ... \cup T^{(k)} \tag{6}$$

Each sub-problem can be formulated as finding a certain f_j with a satisfactory classification rate on $T^{(j)}$:

$$f_j: \quad X \rightarrow T^{(j)} \tag{7}$$

3.2 Parallel Training

With the division of *c* sub-problems, classifiers can construct *c* GA modules and solve them in parallel. Each module is composed of the whole input features and a fraction of the class categories to produce a corresponding fraction of the original problem.

We denote:

$$\overline{T}^{(j)} = T - T^{(j)}, \quad j = 1,2,...,\mathrm{c} \tag{8}$$

which means $\overline{T}^{(j)}$ is the complemented set of $T^{(j)}$. Then, the training set for each module can be represented as:

$$\xi_j = \{(X_q, T_q^{(j)})\}_{q=1}^{M} \cup \{(X_q, \overline{T}_q^{(j)})\}_{q=M+1}^{p} \tag{9}$$

where we assume there are *M* instances in the training set whose classes belong to $T^{(j)}$, and the rest belong to $\overline{T}^{(j)}$.

Therefore, with the training of each module, GA in module *j* has two objectives. It will not only classify the data with the class in $T^{(j)}$ correctly, but also ensure that training data for the class(es) in $\overline{T}^{(j)}$ will not be wrongly classified into the class in $T^{(j)}$. In other words, for those class(es) in $\overline{T}^{(j)}$, GA will only distinguish them from the class in $T^{(j)}$. As a result, the GA in each module will converge more quickly.

3.3 Integration

Although each GA module has evolved a portion of the solution, we cannot just simply aggregate their sub-solutions as the final one. As discussed earlier, each GA module only classifies the class in $T^{(j)}$, but not the class(es) in $\overline{T}^{(j)}$. Therefore, when the sub-solutions are combined together, there may still exist conflicts among the sub-solutions. For example, rules from different modules may classify an instance into several classes. In order to resolve these conflicts and further improve the classification rate, the classifier employs some intelligent decision rules. The detailed integration process is explained as follows.

- The classifier constructs an overall rule set by aggregating all rules from *c* modules.
- Some decision rules are added to help resolve the above-mentioned conflicts. We believe that the ending classification rate obtained from each module would be useful for this purpose. Currently, the following decision rules have been employed:
  i) If an instance is classified into more than one class categories by the rule set, it will be classified into the class whose corresponding module achieves the highest classification rate in the parallel training phase, if available.
  ii) If an instance is not classified into any class category by the rule set, it will be classified into the class whose corresponding module achieves the lowest classification rate in the parallel training phase, if available.

4. Relative Importance Factor (RIF) Feature Selection

Fisher's linear discriminant (FLD) algorithm projects an *n*-dimensional feature space to a *c-1* dimensional feature space by the function $y_i = w^t x_i$, in the direction *w* that maximizes the criterion function $J(w) = \frac{w^t S_B w}{w^t S_W w}$, where S_B is called as the between-class scatter matrix, and S_W the within-class scatter matrix (Duda and Hart, 2000).

As we aim to employ a feature selection technique in each class module which only distinguishes two classes, i.e., $T^{(j)}$ and $\overline{T}^{(j)}$, the projected feature space is one-dimensional (projected on one line) in this situation. Hence, the transformation matrix *w* that maximizes the criterion function *J(w)* is a vector $w = [w_1 \quad w_2 \quad \ldots \quad w_n]^t$. The elements in the transformation vector *w* can be viewed as weights for different features in the original feature space respectively. Thus, we can simplify the feature selection technique based on one observation: in an optimal transformation vector *w* of the Fisher's linear discriminant, a larger w_i means that the *ith* feature is more likely to be irrelevant to the module and a smaller w_i means the *ith* feature is less likely to be relevant to the module. This observation forms the basis of the proposed RIF technique.

However, the weights obtained directly from the transformation vector *w* are not normalized. In order to derive a common feature selection metric across different sets of features in different problems, we propose a *Relative Importance Factor (RIF)*, $r = [r_1 \quad r_2 \quad \ldots \quad r_n]^t$, instead of using the transformation vector *w* directly for feature selection. The RIF is obtained through the following two steps (Guan and Li, 2002b):

I. Normalize the length of the transformation vector *w*.
Since we are evaluating the relative importance of features, we are more interested in the relative weights of the features formed from the transformation vector *w*, which can be obtained through normalization:

$$w' = \frac{w}{\sqrt{\sum_{i=1}^{n} (w_i)^2}} \tag{10}$$

where w_i is the weight of the i_{th} feature in w, w' is the normalized transformation vector, and n is the number of features.

II. Render the importance factor independent from the number of features.
Since different problems have different numbers of features in their feature spaces, it is necessary to make the RIF values independent of the number of features in the feature space. This is achieved by the following function:

$$r = \frac{n}{\sum_{i=1}^{n} |w_i'|} w' \tag{11}$$

Combining (10) and (11), RIF values can be obtained from the transformation vector w directly as:

$$r = \frac{n}{\sum_{i=1}^{n} \left| \frac{w_i}{\sqrt{\sum_{i=1}^{n} (w_i)^2}} \right|} * \frac{w}{\sqrt{\sum_{i=1}^{n} (w_i)^2}} = \frac{n}{\sum_{i=1}^{n} |w_i|} w \tag{12}$$

The elements of r represent the normalized importance of different features, which are independent from the magnitude of w and the number of features in the feature space.

The proposed RIF technique requires much less computation time. Assume there are n input features in the original feature space. In order to obtain the relative importance of each feature, n FLD computations with $n-1$ features included is needed each time using traditional knock-out techniques. With our simplified method, the relative importance of each feature in the module (RIF value) can be obtained in one computation with all n features included.

RIF values are used as the feature selection tool in our modular GA-based classification. The feature selection technique can be summarized as follows:

Step 1: Calculate the Fisher's transformation vector w with respect to all features in the input feature space for each class module.
Step 2: Calculate the RIF value for each feature by using formula (12).
Step 3: Set a threshold value *T1*. If the RIF value of a feature is less than *T1*, it can be considered as an irrelevant feature (IRF).
Step 4: Remove IRFs from each module. A new set of features for each class module will be selected.
Step 5: Modular GA-based classification is then performed based on the new feature set for each class module, as presented in Section 3.

5. Experimental Results and Analysis

5.1 Experimental Scheme

We have implemented several classifiers running on three benchmark data sets to evaluate our approaches. The data sets chosen are the wine data, glass data, and diabetes data. The first two are available in the UCI machine learning repository (Blake and Merz, 1998), and the last one is taken from the PROBEN1 collection (Prechelt, 1994). They all are real-world problems.

We partition each data set into two parts with an equal number of instances. One half is for training, and the other half is for testing. We use the training data to train the rule set, and test the generalization power of resulting rule set with the test data.

All experiments are completed on Pentium III 650MHz PCs. The results reported are averaged over ten independent runs. The parameters, such as mutation rate, crossover rate, generation limits, are given under the results. We record the evolution of each module and the integration process, but we are only interested in some indicative metrics, which include initial classification rate (CR), generation cost, training time, ending CR, and test CR. The CR in each generation is the best rate achieved by the whole population.

We follow the five steps listed in the last section to determine the IRFs and evaluate the performance of classifiers with those IRFs removed. Then, by comparing to the performance of a

classifier with the complete feature set, it can be shown whether the performance of our modular classifiers have improved or degraded as a result of removing IRFs.

5.2 The Wine Data

The wine data contains the chemical analysis of 178 wines from three different cultivars in the same region in Italy. The analysis determines the quantities of 13 constituents found in each of the three types of wines. In other words, it has 13 continuous attributes, 3 classes, 178 instances, and no missing values.

Table 1. RIF value for each feature in different class modules - wine data

RIF	Class=1	Class=2	Class=3
Feature 1	1.8773	1.0938	0.9735
Feature 2	0.1974	0.4703	0.4643
Feature 3	3.9760	3.1118	2.8734
Feature 4	0.4459	0.1332	0.1047
Feature 5	**0.0026**	**0.0005**	**0.0014**
Feature 6	0.9748	**0.0398**	0.1324
Feature 7	1.7762	1.4848	1.6627
Feature 8	0.1335	2.8563	2.8634
Feature 9	0.6493	0.2832	0.2381
Feature 10	0.1636	0.5319	0.5601
Feature 11	0.9828	2.6315	2.6018
Feature 12	1.8124	0.3605	0.5223
Feature 13	**0.0082**	**0.0023**	**0.0017**

Notes:
1. Each row in the table records the RIF value for each feature under each class module;
2. The threshold value is chosen as T1=0.1; those values below the threshold are highlighted.

Table 1 shows the RIF value for each feature in different class modules. If we set the threshold value as T1=0.1, feature 5 and 13 are regarded as common IRFs in class module 1, 2 and 3, while feature 6 is regarded as an IRF in class module 2 only. Therefore, feature 5 and 13 are removed from the feature set for module 1 and 3, and feature 5, 6, and 13 are removed from the feature set for module 2. Table 2 shows the comparison of the classifier performance with/without feature selection on the wine data. We can find that the test CRs are improved in all modules as a result of removing all IRFs. For example, the test CR of module 2 gets an improvement from 0.8371 to 0.8657 by 3.4%. In addition, we can also find that the overall test CR is improved with an increase from 0.8652 to 0.8831 by 2.1%.

We also notice that the number of generations and training time needed for each module become either shorter (for module 3) or longer (for module 1 and 2), after the IRFs are removed. This means that the classifier with a reduced feature set either converges quickly or needs more generations to reach a higher performance. Furthermore, module 2 obtains the largest improvement, which is mainly due to the removal of three features.

5.3 The Glass Data

The glass data set contains data of different glass types. The results of a chemical analysis of glass splinters (the percentage of eight different constituent elements) plus the refractive index are used to classify a sample to be either float processed or non-float processed building windows, vehicle windows, containers, tableware, or head lamps. This data set consists of 214 instances with 9 continuous features from 6 classes.

Table 3 shows the RIF values for each feature in different class modules. We choose the threshold value as 0.1, and find that different features can be regarded as IRFs in different class modules, as highlighted in the table. Therefore, we remove all IRFs from each class module. The performance of the classifier trained with the complete set of features and the one with IRFs removed are shown in Table 4 and 5 respectively.

Table 2. Performance of the classifier with/without feature selection - wine data

		Module 1 (Class=1)	Module 2 (Class=2)	Module 3 (Class=3)
Using All Features	Initial CR	0.8876	0.7618	0.8685
	Generations	23.3	48.6	38.5
	T. time (s)	31.2	64.3	50.1
	Training CR	0.9989	1.0	0.9921
	Test CR	0.9033	0.8371	0.8703
		Integration		
	Training CR	0.9966		
	Test CR	0.8652		
Removing al IRFs from each module	Initial CR	0.8899	0.7787	0.8708
	Generations	42.7	55.7	32.1
	T. time (s)	51.7	68.1	39.7
	Training CR	0.9944	0.9933	0.9955
	Test CR	0.9067	0.8657	0.8833
		Integration		
	Training CR	0.9933		
	Test CR	0.8831		

Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%.
2. For each module, ruleNumber=2, popSize=50, generationLimit=100, stagnationLimit=30.
3. "Initial CR" means the best classification rate achieved by the initial population on the training data.
   "Generations" means the generation needed to reach the stopping criteria.
   "T. time (s)" means the training time cost, and its unit is second.
   "Training CR" means the best classification rate achieved by the resulting population on the training data.
   "Test CR" means the best classification rate achieved by the resulting population on the test data.
4. The following tables regarding the performance of our classifier follow the same notation as noted in this table.

Table 3. RIF value for each feature in different class modules - glass data

RIF	Class=1	Class=2	Class=3	Class=4	Class=5	Class=6
Feature 1	6.5212	6.8322	8.9045	6.8777	8.0957	8.2007
Feature 2	0.3349	0.3327	**0.0066**	0.2782	0.2675	0.1111
Feature 3	0.3951	0.2882	**0.0002**	0.2596	**0.0984**	**0.0981**
Feature 4	0.2575	0.2605	**0.0288**	0.2204	**0.0311**	**0.0843**
Feature 5	0.3708	0.3064	**0.0231**	0.2289	0.1945	0.1293
Feature 6	0.3709	0.3183	**0.0106**	0.5747	0.1378	0.1062
Feature 7	0.3427	0.3011	**0.0064**	0.3070	0.1389	**0.0815**
Feature 8	0.3656	0.3241	**0.0040**	0.2076	**0.0158**	0.1826
Feature 9	**0.0413**	**0.0365**	**0.0156**	**0.0458**	**0.0204**	**0.0062**

Notes:
1. Each row in the table records the RIF value for each feature under each class module;
2. The threshold value is chosen as T1=0.1; those values below the threshold are highlighted.

Comparing the corresponding module elements in Table 4 and 5, we can find that the ending CR for each module is either improved or degraded slightly after IRFs are removed from the six modules, i.e., the test CRs of module 1, 2, and 3 have improved, while the test CRs of module 4, 5, 6 have degraded. However, the overall test CR is still improved from 0.4224 to 0.4944 (17%) after the integration process. This tells us that that removing IRFs may result in performance deterioration in some modules, which also means the selection of a suitable threshold is crucial, but it may still be beneficial to the overall performance.

Table 4. Performance of the classifier with the complete set of features - glass data

	Module 1 (Class=1)	Module 2 (Class=2)	Module 3 (Class=3)	Module 4 (Class=4)	Module 5 (Class=5)	Module 6 (Class=6)
Initial CR	0.7308	0.7224	0.9187	0.9523	0.9664	0.9561
Generations	125.2	128.1	67.0	50.1	30.4	42.0
T. time (s)	89.6	93.7	40.5	29.7	18.2	32.7
Training CR	0.9421	0.9178	0.9299	0.9944	0.9963	0.9972
Test CR	0.6776	0.6196	0.8832	0.9299	0.9411	0.9449
	Integration (Class=1, 2, 3, 4, 5, 6)					
Training CR	0.7738					
Test CR	0.4224					

Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%;
2. For each module, ruleNumber=5, popSize=100, generationLimit=150, stagnationLimit=50;

Table 5. Performance of the classifier with all IRFs removed - glass data

	Module 1 (Class=1)	Module 2 (Class=2)	Module 3 (Class=3)	Module 4 (Class=4)	Module 5 (Class=5)	Module 6 (Class=6)
Initial CR	0.7346	0.7121	0.9150	0.9570	0.9636	0.9701
Generations	127.5	123.8	50.1	24.8	59.3	39.0
T. time (s)	87.3	83.9	33.6	14.2	34.0	29.0
Training CR	0.9243	0.9075	0.9160	0.9991	0.9822	0.9953
Test CR	0.7056	0.6785	0.9243	0.9168	0.9234	0.9252
	Integration (Class=1, 2, 3, 4, 5, 6)					
Training CR	0.7720					
Test CR	0.4944					

Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%;
2. For each module, ruleNumber=5, popSize=100, generationLimit=150, stagnationLimit=50;

5.4 The Diabetes Data

The diabetes problem diagnoses diabetes of Pima Indians. It has 8 features, 2 classes, and 768 instances. All features are continuous, and they are number of times pregnant, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, 2-hour serum insulin, body mass index, diabetes pedigree function, and age.

We still use the RIF values to determine the IRFs of the diabetes data. Since the diabetes data have only 2 classes, each feature has the same RIF value in the two class modules as shown in Table 6. If the threshold is chosen as T1=0.1, feature 4 is regarded as the IRF for both modules.

Table 6. RIF value for each feature in different class modules - diabetes data

RIF	Class=1/ Class=2
Feature 1	0.8291
Feature 2	2.8045
Feature 3	0.6738
Feature 4	**0.0366**
Feature 5	0.3618
Feature 6	2.1049
Feature 7	0.8168
Feature 8	0.3725

Notes:
1. Each row in the table records the RIF value for each feature under each class module;
2. The threshold value is chosen as T1=0.1; those values below the threshold are highlighted.

Table 7. Performance of the classifier with different set of features - diabetes data

Using All Features	Module 1 (Class=1)	Module 2 (Class=2)
Initial CR	0.6966	0.6852
Generations	179.1	186.4
T. time (s)	366.5	358.9
Training CR	0.8542	0.8234
Test CR	0.7336	0.7279
	Integration (Class=1, 2)	
Training CR	0.8388	
Test CR	0.7365	

Removing Feature 4	Module 1 (Class=1)	Module 2 (Class=2)
Initial CR	0.6958	0.7047
Generations	179.4	159.7
T. time (s)	387.6	316.6
Training CR	0.8552	0.8167
Test CR	0.7349	0.7385
	Integration (Class=1, 2)	
Training CR	0.8411	
Test CR	0.7477	

Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%;
2. For each module, ruleNumber=15, popSize=100, generationLimit=200, stagnationLimit=30;

We remove feature 4 from both modules and the resulting performance of our classifier is reported in Table 7, which compares the classifier performance under two scenarios, i.e., the special case when feature 4 is removed and the normal case when all features are used for classification. We notice that test CRs are improved for both modules, and training CR is improved for module 1. As for the final training CR and test CR, they all are improved after feature 4 is removed from the feature set. These results on the diabetes data have again shown that the effect of removing IRFs successfully reduces the feature space dimension and helps improve the classifier performance.

As the diabetes data has only two classes, and feature 4 is the common IRF for both class modules, a general non-modular GA approach with RIF feature selection technique is also feasible. An experiment with the non-modular GA approach has been conducted to contrast with the modular GA approach, and the results are shown in Table 8.

Table 8. Performance of the non-modular GA classifier - diabetes data

	Using All Features	**Removing Feature 4**
Initial CR	0.6273	0.6393
Generations	178.1	184.0
T. time (s)	581.6	616.8
Training CR	0.7568	0.7747
Test CR	0.6961	0.7289

Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%;
2. ruleNumber=30, popSize=100, generationLimit=200, stagnationLimit=30;

We still find that removing feature 4 (IRF) improves both the training CR (from 0.7568 to 0.7747 by 2.4%) and test CR (from 0.6961 to 0.7289 by 4.7%) with the non-modular GA approach. If we compare the corresponding results of these two approaches, it is shown that the performance of the non-modular approach is inferior to that of the modular approach in terms of the final training CR and test CR, which shows that class decomposition approach can improve the classifier performance.

5.5 Reduction in Rule Set Complexity

As the IRFs are removed from the feature space, the resulting rule sets for a classification problem become shorter and more concise, i.e, the complexity is reduced. When these reduced rule sets are used to classify data, it is apparent that the classifier can achieve a higher throughput. We count the total number of attribute genes to evaluate the improvement on complexity. The lists in the Appendix show the reduction of rule set complexity on the diabetes data. By comparing the number of genes in each rule set, we find the complexity is reduced by 10% (c.f. the Appendix). We also measure the rule set complexity for the wine and glass data. As a result, a reduction rate of 17.9% and 25% is achieved for the wine and glass data respectively. We find that the rule set for glass data achieves the highest reduction rate, as more IRFs are removed from the feature space.

5.6 Comparison to the Application of RIF in Neural Networks

RIF has also been applied successfully to neural networks for feature selection. The results were reported in (Guan and Li, 2002b). GA and neural network are two different soft computing techniques, both having advantages and shortcomings. The knowledge extracted by neural network is hidden and distributed over the network, while GA has comparatively more explanatory power, as it explicitly shows the evolutionary process of solutions and the solution format is decodable.

In (Guan and Li, 2002b), the diabetes data are also used to test the effect of RIF in neural networks. When feature 4 (IRF) is removed, the classifier achieves a classification error of 23.96% on the test data (25% of the whole data), which is equal to a test CR of 0.7604. According to the results reported in Table 7, the modular GA-approach with RIF achieves a test CR of 0.7477 on the test data (50% of the whole data). In order to be fair, we also use the 25% of the whole data as test data, and achieve a test CR of 0.7531. We can conclude that the performance of our modular GA approach with RIF is comparable to that of neural networks.

6. Conclusions and Discussions

This paper proposes a new feature selection technique, Relative Importance Factor (RIF), to find irrelevant features in the input domain of a classification problem. By removing these features, we aim to improve classification accuracy and reduce the dimensionality of the classification problems. RIF is employed in modular GA-based classifiers. In this modular approach, a classification problem is decomposed into several modules in terms of class decomposition, and each module is responsible for solving a fraction of the original problem. These modules can be trained in parallel, and the sub-solutions obtained from them are integrated to form the final solution. RIF is used as a feature selection technique to detect the IRFs in each class module.

Three benchmark classification data sets have been used to evaluate the proposed approaches. The experiment results show that RIF can be used as a simple and yet effective feature selection technique to determine irrelevant features and help achieve higher classification accuracy with the feature space dimension reduced. In the meantime, the complexity of the resulting rule sets is also reduced which means the modular classifiers with IRFs removed will be able to classify data with a higher throughput.

The integration of RIF feature selection with a modular GA approach brings forth some advantages. First, as each module is only responsible for one class, it is easier to use RIF values to find the IRFs in that particular class, eliminating the interference from other classes. Second, RIF require relatively small computation cost compared to other feature selection techniques such as the knock-out technique. It is based on the statistic distribution of features in the input feature space and needs only one calculation of FLD transformation weights. Furthermore, RIF is independent from the learning algorithms, and it can also be used with other soft computing techniques such as neural network and other types of classifiers such as Bayes classifiers.

The selection of a good threshold value for RIF is an important issue. In most cases, if we use a larger threshold value, more features can be removed and complexity can be further reduced. However, too large a threshold value may induce information loss, so that classification accuracy can be affected.

The feature selection techniques presented in the paper have different effects on the training and test performance of the classifiers tested. From the experiment results, we find that training CR sometimes degrades a little while test CR improves. However, test CR generally improves more than training CR degrades. We focus more on test performance because it represents the generalization capability of a classifier, our results show that it is worth using the proposed feature selection techniques to reduce the feature space dimension.

In this paper, our classifiers partition the output classes in a non-overlapping manner, which means each module only tackles one class. Alternatively, they can have some degrees of overlapping in class decomposition for redundancy or validation purpose. Accordingly, RIF may need a modification to accommodate this overlapping situation, and the selection of the threshold values becomes more significant. We are still researching on these issues.

Appendix. Rule Set Samples for the Diabetes Data

The following two lists show the resulting rule sets for class module 1 of the diabetes data before and after feature selection respectively - removing feature 4 (cf. Table 6 and 7). We can see that feature 4 (X4 in the rule set) does not appear in the second table, as it has been removed from the feature space. If we count the number of genes in each rule set, we find it is reduced from 50 genes in the first rule set

to 45 genes in the second one, with a reduction rate as 10%. Therefore, with a reduced feature space, the rule set complexity is also reduced.

(Rule set for module 1 with all features)

```
1. IF (0.81<=X2<=1.01) THEN Class=1
2. IF (0.71<=X6<=0.88) AND (0.07<=X8<=0.51) THEN Class=1
3. IF (0.53<=X6<=0.59) AND (0.36<=X7<=0.88) THEN Class=1
4. IF (0.45<=X2<=0.47) AND (0.49<=X3<=0.64) AND (0.61<=X4<=0.70) THEN Class=1
5. IF (0.43<=X2<=0.63) AND (0.28<=X3<=0.83) AND (0.22<=X4<=0.40) AND
       (0.45<=X6<=0.87) AND (0.01<=X7<=0.42) AND (0.12<=X8<=0.32) THEN Class=1
6. IF (0.00<=X1<=0.12) AND (0.83<=X2<=0.91) AND (0.38<=X3<=0.99) AND
       (0.46<=X4<=0.46) AND (0.46<=X5<=0.55) AND (0.14<=X6<=0.70) AND
       (0.46<=X7<=0.75) THEN Class=1
7. IF (0.03<=X1<=0.76) AND (0.79<=X2<=0.92) AND (0.18<=X4<=0.86) THEN Class=1
8. IF (0.16<=X3<=0.64) AND (0.32<=X4<=0.47) AND (0.57<=X7<=0.65) THEN Class=1
9. IF (0.38<=X1<=0.87) AND (0.45<=X2<=1.00) AND (0.18<=X7<=0.39) THEN Class=1
10. IF (0.18<=X5<=0.53) AND (0.48<=X7<=0.64) THEN Class=1
11. IF (0.59<=X2<=0.78) AND (0.41<=X6<=0.84) AND (0.35<=X8<=0.64) THEN Class=1
12. IF (0.50<=X2<=0.75) AND (0.89<=X3<=0.91) AND (0.71<=X6<=0.95) AND
       (0.44<=X7<=0.72) THEN Class=1
13. IF (0.14<=X1<=0.42) AND (0.41<=X7<=1.01) THEN Class=1
14. IF (0.09<=X1<=0.30) AND (0.57<=X4<=0.60) AND (0.70<=X5<=0.73) AND
       (0.10<=X6<=0.34) AND (0.35<=X7<=0.37) THEN Class=1
15. IF (0.44<=X3<=0.82) AND (0.24<=X5<=0.54) AND (0.30<=X7<=0.72) AND
       (0.36<=X8<=0.83) THEN Class=1
```

(Rule set for module 1 with feature selection – feature 4 is removed)

```
1. IF (0.59<=X2<=0.98) AND (0.27<=X3<=0.71) AND (0.47<=X7<=0.58) THEN Class=1
2. IF (0.36<=X1<=0.55) AND (0.70<=X2<=0.97) AND (0.54<=X3<=0.91) THEN Class=1
3. IF (0.12<=X3<=0.96) AND (0.18<=X5<=0.51) AND (0.38<=X8<=0.75) THEN Class=1
4. IF (0.14<=X2<=0.97) AND (0.82<=X3<=0.97) AND (0.62<=X5<=1.00) AND
       (0.90<=X7<=0.91) AND (0.27<=X8<=0.93) THEN Class=1
5. IF (0.22<=X3<=1.00) AND (0.24<=X5<=0.33) AND (0.33<=X6<=0.73) AND
       (0.25<=X7<=0.80) AND (0.11<=X8<=0.30) THEN Class=1
6. IF (0.54<=X2<=0.68) AND (0.39<=X6<=1.00) AND (0.20<=X8<=0.59) THEN Class=1
7. IF (0.46<=X2<=0.95) AND (0.58<=X5<=0.91) THEN Class=1
8. IF (0.57<=X1<=0.91) AND (0.03<=X2<=0.17) AND (0.59<=X5<=0.75) THEN Class=1
9. IF (0.65<=X3<=0.78) AND (0.27<=X7<=0.42) AND (0.24<=X8<=0.99) THEN Class=1
10. IF (0.31<=X1<=0.43) AND (0.37<=X3<=0.73) AND (0.42<=X5<=0.50) AND
       (0.27<=X7<=0.29) AND (0.15<=X8<=0.93) THEN Class=1
11. IF (0.02<=X6<=0.11) THEN Class=1
12. IF (0.81<=X2<=1.00) THEN Class=1
13. IF (0.48<=X6<=0.50) AND (0.04<=X7<=0.70) AND (0.37<=X8<=0.69) THEN Class=1
14. IF (0.71<=X1<=0.76) AND (0.86<=X3<=0.95) AND (0.30<=X6<=0.34) THEN Class=1
15. IF (0.43<=X1<=0.70) AND (0.09<=X5<=0.73) THEN Class=1
```

References

1. R., Anand, Mehrotra, K., Mohan, C.K., Ranka, S., 1995. Efficient classification for multiclass problems using modular neural networks. IEEE Transactions on Neural Networks, 6 (1), pp. 117-124.
2. Battiti,. R. 1994. Using mutual information for selecting features in supervised neural net learning. IEEE Transaction on Neural Networks, 5 (4), pp. 537-550.
3. Blake, C.L., Merz, C.J., 1998. UCI Repository of machine learning databases (http://www.ics.uci.edu/~mlearn/MLRepository.html). Irvine, CA: University of California, Department of Information and Computer Science.
4. Corcoran, A.L. and Sen, S., 1994. Using real-valued genetic algorithm to evolve rule sets for classification. Proceedings of the 1st IEEE Conference on Evolutionary Computation, Orlando, US, pp. 120-124.
5. Duda, R.O., Hart, P.E., and Stork, D.G. 2000, Pattern Classification, New York: Wiley, 2nd Edition.
6. Fukunaga, K., 1990, Introduction to Statistical Pattern Recognition, 2nd ed., Boston: Academic Press.
7. Gonzalez, A. and Perez, R. 2001. Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Transaction on Systems, Man and Cybernetics, Part B, 31 (3), pp. 417-425.
8. Guan, S.U. and Li, S.C., 2002. Parallel growing and training of neural networks using output parallelism. IEEE Transactions on Neural Networks, 13 (3), pp. 1-9.

9. Guan, S.U. and Li, P, 2002. Feature selection for modular neural network classifiers, to appear in Journal of Intelligent Systems.
10. Ishibuchi, H., Nakashima, T., Murata, T, 1999. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Transactions on Systems, Man and Cybernetics, Part B, 29 (5), pp. 601-618.
11. Jenkins, R.E., Yuhas, B.P., 1993. A simplified neural network solution through problem decomposition: the case of the truck backer-upper. IEEE Transactions on Neural Networks, 4 (4), pp. 718-720.
12. Kwak, N. Choi, C.H. 2002. Input feature selection for classification problems. IEEE Transaction on Neural Networks, 13 (1), pp. 143-159.
13. Lerner, B., Levinstein, M., Rosenberg, B., Guterman, H., Dinstein, L., Romem, Y. 1994. Feature selection and chromosome classification using a multilayer perceptron neural network. IEEE International Conference on Neural Networks, vol. 6, pp. 3540-3545.
14. Lu, B.L. and Ito, M., 1999. Task decomposition and module combination based on class relations: a modular neural network for pattern classification. IEEE Transactions on Neural Networks, 10 (5), pp. 1244-1256.
15. Merelo, J.J., Prieto, A., Moran, F., 2001. Optimization of classifiers using genetic algorithms. In: Patel, M., Honavar, V., Balakrishnan, K. (Eds.), Advances in the Evolutionary Synthesis of Intelligent Agents. MIT Press, Cambridge.
16. Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed. Springer-Verlag, New York.
17. Pal, S.K., De, R.K., and Basak, J., 2000, Unsupervised feature evaluation: a neuro-fuzzy approach, IEEE Transactions on Neural Networks, 11 (2), March 2000, pp. 366 –376.
18. Prechelt, L., 1994. PROBEN1: A set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, Department of Informatics, University of Karlsruhe, Germany.
19. Setiono, R. and Liu, H., 1997. Neural network feature selector. IEEE Transactions on Neural Networks, 8 (3), pp. 654-662.
20. Setnes, M., Roubos, H., 2000. GA-Fuzzy modeling and classification: complexity and performance. IEEE Transactions on Fuzzy Systems 8 (5), pp. 509-522.
21. Verikas, A. and Bacauskiene M., 2002, Feature selection with neural networks. Pattern Recognition Letters 23, pp.1323-1335.