Learning Discriminant Functions based on Genetic Programming and Rough Sets

BEEN-CHIAN CHIEN*, JUI-HSIANG YANG AND TZUNG-PEI HONG

Department of Computer Science and Information Engineering
National University of Tainan, Tainan, Taiwan 700, R.O.C.
E-mail: bcchien@mail.nutn.edu.tw

Department of Information Engineering, I-Shou University
Kaohsiung County, Taiwan 840, R.O.C.
E-mail: m9003012@isu.edu.tw

Department of Computer Science and Information Engineering
National University of Kaohsiung, Kaohsiung, Taiwan 811, R.O.C.
E-mail: tphong@nuk.edu.tw

Received: December 15, 2009. Accepted: April 20, 2010.

Supervised learning based on genetic programming can find different classification models including decision trees, classification rules and discriminant functions. The previous researches have shown that the classifiers learned by GP have high precision in many application domains. However, nominal data cannot be handled and calculated by the model of using discriminant functions. In this paper, we present a scheme based on rough set theory and genetic programming to learn discriminant functions from general data containing both nominal and numerical attributes. The proposed scheme first transforms the nominal data into numerical values by applying the technique of rough sets. Then, genetic programming is used to learn discriminant functions. The conflict problem among discriminant functions is solved by an effective conflict resolution method based on the distance-based fitness function. The experimental results show that the classifiers generated by the proposed scheme using GP are effective on nominal data in comparison with C4.5, CBA, and NB-based classifiers.

Keywords: Machine learning, discriminant function, genetic programming, classification, rough sets.

1 INTRODUCTION

Classification is one of the important research topics in data mining. The approaches of learning classifiers are also known as supervised learning in
machine learning. The task of supervised learning is to find classifiers from a set of data instances with pre-defined classes called the training set. Unknown class data then can be classified by the learned classifier. Owing to the versatility of human activities and the unpredictability of data, a supervised learning model generally is hard to handle datasets well in all kinds of application domains [17]. Many classification models had been proposed in the past. Some of the previous schemes were based on mathematical models or theories. For example, Naïve Bayes Classifier [5], Bayesian network classifiers [9–10], NBTTree [13] and SNNB [31], are well-known probability-based classification methods based on Bayesian decision theory [10–13]. The decision tree is another widely used classification model represented as a flow-chart-like tree structure. Each internal node of the tree denotes a decision on an attribute, each branch represents an outcome of the decision, and leaf nodes represent classes. ID3 [26] and C4.5 [27] are the famous approaches of decision trees using an entropy-based measure known as information gain to select one of the “decision” attributes in each iteration. Other classification models based on the techniques of modern computational intelligence like data mining and evolutionary computation were applied for developing effective classifiers in recent decade. For example, CBA [19] employs association rules to develop a hybrid rule-based approach for classification. Genetic programming (GP) is a kind of evolutionary computing approach that encodes a set of pre-defined functions to be an expression tree. Then, the expression trees are evolved by evolution operators such as reproduction, crossover and mutation for a number of generations to find the effective solution for satisfying a specified fitness function.

In [7], Espejo et al. briefly divided the extraction of classification models using GP into three different classifiers: decision tree, classification rules, and discriminant functions. Decision trees and classification rules have good interpretability but need more function nodes and evolution time for evolving accurate rules in learning phase. The classification model using discriminant functions can perform fewer evolving steps in the learning phase and the classifiers are efficient and accurate in classification phase [11]. However, only numerical attributes are considered in the function model since the expressions in classification functions are composed of arithmetic operations and mathematical functions. Furthermore, a discriminant function generally accomplishes the problem of binary classification. For multi-class classification problem, we need one discriminant function for each class. While making decision among the multi-class discriminant functions, the rejection occurs if an unknown object is not recognized by any discriminant function and the conflict occurs if an object is recognized by two or more classification functions at the same time. The cases of rejection and conflict decrease the classification accuracy of a classifier. In [11], Kishore proposed the SA measure to solve the conflict problem. However, the SA measure was not improved much under the usage of an accuracy-based fitness function in GP. Another solution of the decision tree based on classification functions
was proposed in [17]. The main drawback of using the decision tree is that an object is assigned to the unknown class if it cannot be recognized by any node of the decision tree. This means that the rejection problem was not really solved by decision tree.

In this paper, a supervised learning scheme based on rough sets and genetic programming is proposed to generate discriminant functions for classification. In the learning phase, we define rough membership functions for nominal attributes based on rough set theory and give a distance-based fitness function for genetic programming to learn a set of discriminant functions. The rough membership function is used to transform nominal attributes into numerical attributes to overcome the problem of calculating mathematical functions. In the classification phase, an effective conflict resolution method based on the distance-based fitness function is developed to resolve the problems of reject and conflict. For demonstrating the effectiveness of the proposed scheme, twenty data sets containing nominal attributes are selected from the UCI data repository [1] for accomplishing the experiments. The experimental results were compared with the approaches that are well-done on categorical data including three Naïve Bayes classifiers [5, 13, 31], the decision tree classifier C4.5 [27] and the classifier CBA [19]. The results show that accurate discriminant functions can be learned to build effective classifiers from general datasets with both numerical and nominal attributes by the proposed scheme.

This paper is organized as follows: Section 2 briefly reviews the supervised learning techniques using GP and rough membership functions based on rough set theory. In Section 3, we discuss the proposed learning algorithm based on the transformation of rough attribute membership and genetic programming. In Section 4, we present the conflict resolution in the classification algorithm. Section 5 shows the experimental results and makes comparisons with other classifiers. Finally, conclusions are drawn in Section 6.

2 REVIEW OF RELATED RESEARCH

2.1 Genetic Programming for Learning Classification Models

The technique of genetic programming was proposed by Koza [14–15] in 1987 and has been applied to solve many problems such as symbolic regression, robot control programs and classification. Genetic programming was used by many researchers to develop different classification models in the past decades. Generally, three different classifiers, decision tree, classification rules, and discriminant functions, can be learned and constructed using genetic programming [7].

While learning classification models by GP, a decision tree is a natural structure since the representation of genes in GP is constructed as a tree. Each internal node in the decision tree model tests the value of one of the
attributes. The branches of an internal node are labeled with the results of the test on the attribute. Each leaf node stands for a specific class and a class label is assigned to it [6, 28]. The classification model (classifier) is the evolved final decision tree.

Although the final decision tree can be interpreted as a set of classification rules, the structure of nodes for learning classification rules by GP is different from decision trees [2–8, 30–32]. The antecedent part of a classification rule contains logical connectives like conjunction, disjunction and negation, which are used to be the nodes of an expression tree in GP’s population. The consequent part is the class satisfying the conditions of the expression tree. For classifying multiclass problem, a set of classification rules should be learned to build a classifier.

Discriminant functions are mathematical expressions in which the attributes of instances are combined with different operators. The calculating result of an expression is a numerical value located in a specified range indicating the class predicted. A discriminant function can be trained to classify a binary classification problem [2] or a multiclass problem [20]. The binary discriminant functions also can be used to solve the multiclass problem. This approach needs involve an individual discriminant function for each class in the classifier [3–4, 11–12]. Thus, a classifier is composed of a set of discriminant functions. Learning discriminant functions for classification by GP need to evolve expression trees with a set of arithmetic operators or mathematical functions and a qualified fitness function. Since the expression accepts numerical values only, the previous classifiers using discriminant functions cannot handle datasets with nominal attributes containing categorical data.

2.2 Rough Sets and Rough Membership Functions
The mathematical foundation of rough set theory is based on the set approximation of partition space on sets. In this section, we briefly review some preliminaries of rough set theory that are relevant to this research. For more details, one may refer to [24] and [25].

The idea of rough sets is based on the establishment of equivalence classes on a given data set $S$. A rough set definition for a given class $X$ is approximated by two sets of equivalence classes called a lower approximation of $X$ and an upper approximation of $X$. The lower approximation of the class $X$ contains the equivalence classes that are certain to belong to $X$ without ambiguity. The upper approximation of the class $X$ contains the equivalence classes that cannot be described as the space not belonging to $X$. In this section, we use the formal definition in [24] to introduce rough sets. We first define an information system as follows:

**Definition 1:** Let $U = (S, A)$ be an information system where $S$ is a non-empty, finite set of objects called the universe and $A$ is a non-empty, finite set
of attributes. For each $B \subseteq A$, $a \in A$, there is an equivalence relation $E_A(B)$ such that

$$E_A(B) = \{(x, x') \in S^2 | \forall a \in B, a(x) = a(x')\}.$$ 

If $(x, x') \in E_A(B)$, we say that objects $x$ and $x'$ are indiscernible by the attributes in $B$. $E_A(B)$ is called $B$-indiscernibility relation.

An attribute-value table can be an information system in which the objects of the universe are values in rows and the attributes are columns. A classification information system generally can be viewed as a table with different attributes of features and one distinguished attribute called the decision attribute. For example, Table 1 describes an information system containing 14 objects in rows and 5 attributes including the decision attribute ‘play’. Thus, the universe $S = \{x_1, x_2, \ldots, x_{14}\}$ and the set of attributes $A = \{\text{outlook, temperature, humidity, windy, play}\}$, where ‘play’ is the decision attribute.

**Definition 2:** $apr = (S, E)$, is called an approximation space. The object $x \in S$ belongs to one and only one equivalence class. Let $[x]_B = \{y | x \in E_A(B) y, \forall x,$

<table>
<thead>
<tr>
<th>objects</th>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>$x_2$</td>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>$x_3$</td>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>$x_4$</td>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>$x_5$</td>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>$x_6$</td>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>$x_7$</td>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>$x_8$</td>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>$x_9$</td>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>$x_{14}$</td>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
</tbody>
</table>

**TABLE 1**
The weather data set [21]
\[ y \in S \}, \text{ and } [S]_B = \{ [x]_B \mid x \in S \}. \text{ The notation } [x]_B \text{ denotes equivalence classes of } E_A(B) \text{ and } [S]_B \text{ is a partition denoting the set of all equivalence classes } [x]_B \text{ for } x \in S. \]

The approximation space basically is a partition based on a set of attributes. For example, let’s consider the set of attributes \( B = \{ \text{outlook, temperature} \} \) in the information system of Table 1. The relation \( E_A(\{ \text{outlook, temperature} \}) \) defines the following partition of the universe

\[ [S]_{\{ \text{outlook, temperature} \}} = \{ \{ x_1, x_2 \}, \{ x_3, x_{13} \}, \{ x_4, x_{10}, x_{14} \}, \{ x_5, x_6 \}, \{ x_7 \}, \{ x_8, x_{11} \}, \{ x_9 \}, \{ x_{12} \} \}. \]

Assume that \( X \) is one of the classes in the decision attribute of the information system; the two approximations on the class \( X \), the lower approximation \( BX \) and the upper approximation \( \overline{BX} \), are defined as follows:

**Definition 3:** Let \( U = (S, A) \), a class \( X \subseteq S \) and \( B \subseteq A \), \( BX \) is lower approximation of \( X \), iff

\[ BX = \{ x \mid x \in S, [x]_B \subseteq X \}. \]

\( \overline{BX} \) is upper approximation of \( X \), iff

\[ \overline{BX} = \{ x \mid x \in S \text{ and } [x]_B \cap X \neq \emptyset \}. \]

For example, let the decision attribute in Table 1 be ‘play’ and \( X \) be the concept of the class ‘yes’. If \( B = \{ \text{outlook, temperature} \} \), we have

\[ BX = \{ x_3, x_{13}, x_7, x_9, x_{12} \}, \]

\[ \overline{BX} = \{ x_3, x_{13}, x_4, x_{10}, x_{14}, x_5, x_6, x_7, x_8, x_{11}, x_9, x_{12} \}. \]

A description for the vague class \( X \) that cannot be classified as a concept with absolute certain description may contain boundary-line objects. We can think of such uncertain boundary-line objects as a membership degree of an object \( x \) belonging to a class \( X \). The following definition of rough membership is to describe the membership degree of an object \( x \) belonging to a class \( X \) under the partition \([S]_B\) on a specified set of attributes \( B \) [25].

**Definition 4:** For a given class \( X \subseteq S \), a rough membership function of object \( x \) belonging to the class \( X \) on the set of attributes \( B \) is defined as

\[ \mu_B^X(x) = \frac{|[x]_B \cap X|}{|[x]_B|}, \]

(1)
Learning Based on GP & RSS

where \( |[x]_B| \) denotes the cardinality of equivalence classes of \([x]_B\) and \( |[x_B \cap X]| \) denotes the cardinality of the set \([x]_B \cap X\). The value of \( \mu^X_B(x) \) is in the range of \([0, 1]\).

The rough membership value \( \mu^X_B(x) \) can be interpreted as the conditional probability that an object \( x \) belongs to the class \( X \), given that the object \( x \) belongs to \([x]_B\). That is, if two different classes \( X_1 \) and \( X_2 \) consist of the same equivalence class \([x]_B\), \( \mu^X_B(x) \) and \( \mu^X_B(x) \) are used to measure the membership degrees of the object \( x \) belonging to the classes \( X_1 \) and \( X_2 \), respectively. For example, as in Table 1, let \( X \) be the set of objects containing the class ‘yes’ in the decision attribute ‘play’. If \( B = \{\text{outlook}\} \), we have

\[
[S]_{\{\text{outlook}\}} = \{\{x_1, x_2, x_8, x_9, x_{11}\}, \{x_3, x_7, x_{12}, x_{13}\}, \{x_4, x_5, x_6, x_{10}, x_{14}\}\},
\]

\[
[x_1]_{\{\text{outlook}\}} = \{x_1, x_2, x_8, x_9, x_{11}\},
\]

\[
X = \{x_3, x_4, x_5, x_7, x_9, x_{10}, x_{11}, x_{12}, x_{13}\},
\]

\[
\mu^\text{yes}_{\{\text{outlook}\}}(x_1) = \frac{|\{x_1, x_2, x_8, x_9, x_{11}\} \cap \{x_3, x_4, x_5, x_7, x_9, x_{10}, x_{11}, x_{12}, x_{13}\}|}{|\{x_1, x_2, x_8, x_9, x_{11}\}|} = 0.40.
\]

If \( X \) is the set of objects containing the class ‘no’ in the decision attribute ‘play’ and \( B = \{\text{outlook}\} \), we have

\[
X = \{x_1, x_2, x_6, x_8, x_{14}\},
\]

\[
\mu^\text{no}_{\{\text{outlook}\}}(x_1) = \frac{|\{x_1, x_2, x_8, x_9, x_{11}\} \cap \{x_1, x_2, x_6, x_8, x_{14}\}|}{|\{x_1, x_2, x_8, x_9, x_{11}\}|} = 0.60.
\]

According to the partitions on different attributes, the same object may have distinct membership degrees for different classes. For example, Figure 1 shows two different partitions on the sets of attributes \( B_1 \) and \( B_2 \). Assume that the partition \([S]_{B_1}\) on \( B_1 \) performs a horizontal partition and the partition \([S]_{B_2}\) on \( B_2 \) performs a vertical partition, respectively. The class \( X_1 \) is represented as the light grey areas and the classes \( X_2 \) is represented as the deep grey areas. For a piece of object square \( x \), the membership degrees of classes \( X_1 \) and \( X_2 \) on the partition \( B_1 \) are \( \mu^{X_1}_{B_1}(x) \) and \( \mu^{X_2}_{B_1}(x) \), respectively. The membership degrees of classes \( X_1 \) and \( X_2 \) on the partition \( B_2 \) are \( \mu^{X_1}_{B_2}(x) \) and \( \mu^{X_2}_{B_2}(x) \), respectively. In Figure 1(a), \( \mu^{X_1}_{B_1}(x) \) and \( \mu^{X_2}_{B_1}(x) \) show that \( B_1 \) is not an ideal set of attributes for discerning the classes \( X_1 \) and \( X_2 \), since their membership degrees are vague. However, the membership degrees in Figure 1(b), \( \mu^{X_1}_{B_2}(x) \) and \( \mu^{X_2}_{B_2}(x) \), present clear-cut relationships on the classes \( X_1 \) and \( X_2 \). Thus, the membership degrees produced by attributes \( B_2 \) reflect more useful information than attributes \( B_1 \) in terms of classifying the classes \( X_1 \) and \( X_2 \).
3. THE LEARNING ALGORITHM OF DISCRIMINANT FUNCTIONS

3.1 Notation
We first define the symbols used in this section. Let $S$, $A$ and $\mu^x_i(x)$ be the same definitions in Section 2, and $A$ has $n + 1$ attributes, $A = \{a_0, a_1, \ldots, a_n\}$. The symbols are defined as follows:

- $a_i$: The $i$th attribute in $A$; $a_0$ is the decision attribute.
- $D_i$: The domain of attributes $a_i$.
- $v_{ji}$: The value of object $x_j$ for attribute $a_i$; $v_{ji} \in D_i$, $1 \leq i \leq n$.
- $C_k$: The $k$th predefined class in the decision attribute; $C_k \in D_0$.
- $K$: The number of predefined classes in the decision attribute $a_0$.
- $\bar{a}_i$: The set of attributes after transformation of attribute $a_i$.
- $w_{jk}$: The value of object $x_j$ for the $k$th transformed attribute in the attribute set $\bar{a}_i$.
- $f_k$: The discriminant function for $C_k$.
- $\Omega_{(gen)}$: The population of the generation $gen$ in GP.
- $h_{i}^{(gen)}$: The $i$th individual in the population $\Omega_{(gen)}$ of the generation $gen$.
- $E_{i}^{(gen)}$: The fitness value of GP in the $i$th individual of generation $gen$.

The formal definitions of discriminant functions and a classifier based on discriminant functions are given in Section 3.2. The transformation of nominal attributes is given in Section 3.3. The proposed fitness function and the GP-based learning algorithm are presented in Section 3.4.

3.2 Discriminant functions
Given a dataset $S$ having $n + 1$ attributes $A = \{a_0, a_1, \ldots, a_n\}$ and $D_i$ is the domain of $a_i$, for $0 \leq i \leq n$. An object in $S$ is represented as $(v_{j0}, v_{j1}, \ldots, v_{jn})$, where $v_{ji} \in D_i$. Let $x_j = (v_{j1}, \ldots, v_{jn})$ and $a_0$ be the decision attribute represent-
ing the predefined classes in $D_0 = \{C_1, C_2, \ldots, C_K\}$. We say that $<v_{j0}, x_j>$ is a sample. A training set (TS) is defined to be a set of samples:

$$TS = \{<v_{j0}, x_j> \mid x_j = (v_{j1}, \ldots, v_{jn}), v_{ji} \in D_i, 1 \leq j \leq m\},$$

where $m = |TS|$ is the number of samples in $TS$. Note that only for the attribute $a_0$, the set of classes is the same as the domain ($D_0$) of $a_0$ because its possible values include all the classes. A discriminant function [7] $f_k$ for the class $C_k$ is defined as a function

$$f_k : \mathbb{R}^n \rightarrow \mathbb{R},$$

such that $f_k$ satisfies the following conditions:

$$\begin{cases} f_k(x_j) \geq \alpha, \text{ if } v_{j0} = C_k \\ f_k(x_j) \geq \alpha, \text{ if } v_{j0} \neq C_k \end{cases}$$

(2)

where $1 \leq k \leq K$, $1 \leq j \leq m$ and $\alpha$ is a constant specified by users for determining whether $x_j$ belongs to the class $C_k$. A classifier $F$ based on discriminant functions is then defined as a set of discriminant functions on the set of predefined classes:

$$F = \{f_k : \mathbb{R}^n \rightarrow \mathbb{R}, 1 \leq k \leq K\}.$$

### 3.3 The Transformation of Rough Attributes Membership

The discriminant function defined in Section 3.2 has a limitation: only numerical attributes are allowed in a discriminant function $f_k$. It does not work when a dataset contains nominal attributes. In order to learn discriminant functions using genetic programming without any restriction, we need a mechanism that describes categorical data as numerical values and the result still keeps the relationships of the original dataset.

We first reexamine a value $v_{ji}$ of a numerical attribute $a_i$ in a sample $<C_k, x_j>$. Since the object $x_j$ belongs to the class $C_k$, the numerical value $v_{ji}$ is viewed as a special relationship between $C_k$ and the attribute $a_i$. That is, similar values of $a_i$ may have similar effects on classification. This concept can be appropriately applied to the values of nominal attributes as well. While facing categorical data in nominal attributes for a classification problem, many previous researches also concentrated their attention on finding the relationships between nominal attributes and classes. For example, the conditional probability in Naïve Bayes [5] and the information gain of entropy in decision trees [27]. Thus, to preserve the relationships between nominal attributes and their corresponding classes, the rough membership function introduced in Section 2 is considered here, such that similar values of a transformed
attribute may have similar effects on classification. The neighborhood relationship of similar values can thus be kept. As shown in Figure 1, we know that the membership degree of an object belonging to each class is dependent on which attributes we used for partitioning. The more objects with the same nominal value $v_{ji}$ contained in a class $C_k$, the more possibility that the objects with $v_{ji}$ belong to the class $C_k$. The idea of transformation is to replace the categorical value $v_{ji}$ of the object $x_j$ as $K$ membership degrees of $\mu_{a_i}^{C_k}(x_j)$ for $1 \leq k \leq K$. Since an object $x_j$ in each class $C_k$ has different membership degrees on distinct attributes, the relationship between attributes and classes can be learned from GP and represented as discriminant functions.

The transformation of the training set $TS$ is presented as follows:
For the training set $TS$, let us consider an object $x_j$:

$$x_j = (v_{j1}, v_{j2}, \ldots, v_{jn}), v_{ji} \in D_i, 1 \leq i \leq n.$$  

The goal of transformation is to replace the value $v_{ji}$ of the attribute $a_i$ in the object $x_j$ with a list of numerical values using a new set of attributes $\tilde{a}_i$.

This transformation can be summarized as the following two cases:

1. If $a_i$ is a numerical attribute, we have $\tilde{a}_i = \{a_i\}$. Let $w_{jk}$ be the value after transformation, the value of $w_{jk} = v_{ji}$, $w_{jk} \in D_i$. That is, the original numerical data are reserved.

2. If $a_i$ is a nominal attribute, the original nominal attribute $a_i$ will be transformed into a set of $K$ numerical attributes $\tilde{a}_i = \{a_{i1}, a_{i2}, \ldots, a_{iK}\}$, where $K$ is the number of pre-defined classes in $D_i$. The value of $x_j$ on $a_i$, $v_{ji}$, will be transformed into a list of numerical values $w_{jk}, w_{j(k+1)}, \ldots, w_{j(k+K-1)}$, where $w_{jk} \in D_i = [0, 1]$ for $1 \leq k \leq K$. The values in the list are defined as rough membership functions:

$$w_{jk} = \mu_{a_i}^{C_k}(x_j), w_{j(k+1)} = \mu_{a_i}^{C_k}(x_j), \ldots, w_{j(k+K-1)} = \mu_{a_i}^{C_k}(x_j).$$

The rough membership function $\mu_{a_i}^{C_k}(x_j)$, $1 \leq k \leq K$, is defined as Equation (1) in Section 2.

After the transformation, we get the new set of attributes $\tilde{A}$,

$$\tilde{A} = \bigcup_{i=1}^{n} \tilde{a}_i.$$  

The detailed algorithm of transformation for nominal data in a training set $TS$ is listed as follows:

**Algorithm: Transforming nominal attributes to numerical attributes**

Input: A training set $TS$ with $m$ training data.

Output: A new training set $TS'$.

Step 1: Compute the membership functions of $\mu_{a_i}^{C_k}(x)$, $\mu_{a_i}^{C_k}(x)$, $\ldots$, $\mu_{a_i}^{C_k}(x)$, for $1 \leq k \leq K$ and $1 \leq i \leq n$, as defined in Equation (1).
Step 2: For $1 \leq j \leq m$, repeat Step 3 to Step 6.
Step 3: $k = 1$, $i = 1$.
Step 4: If $a_i$ is a numerical attribute, for $x_j = (v_{j1}, v_{j2}, ... , v_{jn})$, $w_{jk} = v_{ji}$, $k = k + 1$ and $i = i + 1$.
Step 5: If $a_i$ is a nominal attribute, $x_j = (v_{j1}, v_{j2}, ... , v_{jn})$, $w_{jk} = \mu_a^{C_j}(x_j)$, $w_{j(k+1)} = \mu_a^{C_j}(x_j)$, $k = k + K$ and $i = i + 1$.
Step 6: If $i \leq n$, go to Step 4.
Step 7: Let $y_j = (w_{j1}, w_{j2}, ... , w_{jn'})$ for $1 \leq j \leq m$, where $n' = (n - r) + K$. $r$ and $K$ are the number of nominal attributes in $A$. We have

$$TS' = \{<v_{j0}, y_j>| y_j = (w_{j1}, w_{j2}, ... , w_{jn'}), 1 \leq j \leq m\}.$$ 

In the above transformation algorithm, the time complexity from Step 3 to Step 6 needs $O(nKm)$ at most, where $m$ is the amount of training data, $n$ is the number of attributes in $A$ and $K$ is the number of predefined classes in $TS$. $K$ is usually a small constant. However, Step 1 is the most time-consuming in the transformation algorithm. The computation time of rough membership functions depends on the time of set partition. As we know, the time of partition is dominated by the sorting algorithm. Thus, the time complexity of Step 1 is $O(m \log m)$ for each attribute. Since there are $n$ nominal attributes at most in the training set, the time complexity of the algorithm is $O(nm \log m)$.

### 3.4 The Learning Algorithm

While learning discriminant functions using genetic programming, the individuals are randomly generated arithmetic expressions represented as binary trees. Each individual is possibly evaluated as a potential solution by the fitness function. For improving the effectiveness of discriminant functions, we need a qualified fitness function for GP to evaluate individuals and generate effective solutions. A distance-based fitness function is proposed instead of the instance-based fitness function proposed in [11]. The distance-based fitness function is described as follows:

Let us consider a discriminant function $f_k$ of a class $C_k$ and a specified constant $a$. For a positive instance $<v_{j0}, y_j>$, $v_{j0} = C_k$ in the training set $TS'$. We urge that $f_k(y_j) \geq \alpha$; on the contrary, $f_k(y_j) < \alpha$ for a negative instance $<v_{j0}, y_j>$, $v_{j0} \neq C_k$. To achieve the objective of $f_k$, two parameters $p$ and $q$ are defined. Let $p > \alpha$, $q < \alpha$ and $p + q = 2\alpha$. The error of a positive instance is measured by

$$E_p = \begin{cases} 0 & \text{if } v_{j0} = C_k \text{ and } f_k(y_j) \geq \alpha \\ [(p - f_k(y_j))^2] & \text{if } v_{j0} = C_k \text{ and } f_k(y_j) < \alpha \end{cases},$$

and the error of a negative instance is measured by
The fitness value of an individual $h_i$ is then evaluated by the following fitness function:

$$fitness(h_i, TS') = -\sum_{j=1}^{m} (E_p + E_u),$$

where $m$ is the number of training samples in $TS'$, $1 \leq j \leq m$. Since the fitness value of an individual is negative and represents the degree of error between the target function and the individual, the fitness value should be as large as possible. The best fitness value is zero.

A discriminant function can be thought of as a non-linear separation in the training set. The purpose of the non-linear discriminant function is to separate the distance between positive cases and the negative cases as large as possible. The distance between $p$ and $q$ is similar to the support vector in SVM (support vector machine) method. The traditional fitness function counts the number of misclassified cases in the training set. Such a counting fitness function reflects only the accuracy of a discriminant function and it cannot distinguish the error degree from two same evaluated fitness values. The distance-based fitness function tries to evaluate not only the number of error cases but also the degree of error. It is more sensitive than the counting based fitness function in the misclassified cases with big error. The genetic programming thus has more chance to choose better individuals to evolve good individuals in the next generation.

A classifier containing a set of discriminant functions $f_k$, $1 \leq k \leq K$, can be generated by iteratively applying the following learning algorithm to the training set for each class. The learning algorithm for a discriminant functions using GP is described in detail as follows:

**Algorithm: Genetic programming for learning a discriminant function**

Input: The training set $TS$.
Output: A function with the best fitness value.

Step 1: Transform nominal attributes to numerical attributes and generate the new training set

$$TS' = \{<v_{j0}, y_j> | y_j = (w_{j1}, w_{j2}, ..., w_{jn}), 1 \leq j \leq m\}.$$
the initial generation, and \( h_i^{(gen)} \) stands for the \( i \)th individual of the generation \( gen \).

Step 3: Evaluate the fitness value of each individual on the training set.
For all \( h_i^{(gen)} \in \Omega^{(gen)} \), compute the fitness values \( E_i^{(gen)} = \text{fitness}(h_i^{(gen)}, TS') \), where the fitness evaluating function \( \text{fitness()} \) is defined as this section.

Step 4: Does it satisfy the conditions of termination?
If the best fitness value of \( E_i^{(gen)} \) satisfies the conditions of termination \( (E_i^{(gen)} = 0) \) or the \( gen \) is equal to the specified maximum generation, the \( h_i^{(gen)} \) with the best fitness value is returned and the algorithm halts; otherwise, \( gen = gen + 1 \).

Step 5: Generate the next generation of individuals and go to Step 3.
The new population of next generation \( \Omega^{(gen)} \) is generated by the ratio of \( P_r, P_c \) and \( P_m \), and goes to Step 3; \( P_r, P_c \) and \( P_m \) represent the probabilities of genetic operators, reproduction, crossover and mutation, respectively.

4 THE CLASSIFICATION ALGORITHM

After generating the set of discriminant functions, the task of classification becomes straightforward. Since the discriminant functions are mathematical formulas, we calculate the discriminant functions and test if the results are larger than the specified value \( \alpha \). However, a classifier generally cannot recognize all objects correctly in real applications. Except in the case of misclassification, two situations, conflict and rejection, may occur. In the case of conflict an object is recognized by two or more discriminant functions at the same time. In the case of rejection an object cannot be recognized by any discriminant function. For both of these situations, unknown objects cannot be assigned to their corresponding classes. A complete classifier should include effective resolutions to solve the problems of conflict and rejection. Thus, we propose our resolution as follows.

The proposed scheme is based on the distance-based fitness values and the Z-score of statistical test. For the discriminant function \( f_k \in F \) corresponding to the class \( C_k \) and the samples \( <v_j, y_j> \in TS' \) with \( v_j = C_k \), let \( \bar{Y}_k \) be the mean of values of \( f_k(y_j) \), \( 1 \leq j \leq m_k \), where \( m_k \) is the number of samples belonging to the class \( C_k \) in \( TS' \). That is,

\[
\bar{Y}_k = \frac{\sum_{<v_j, y_j> \in TS'} f_k(y_j)}{m_k}, \quad 1 \leq j \leq m_k, 1 \leq k \leq K.
\]

(9)

For each \( \bar{Y}_k \), the standard deviation of the values of \( f_k(y_j) \), \( 1 \leq j \leq m_k \), is defined as
For an unknown object $x$, let $y$ be the datum with all numerical values after the transformation of $x$ using rough membership functions. The $Z$-scope of datum $y$ for a discriminant function $f_k$ is defined as

$$Z_k(y) = \left| \frac{f_k(y) - \bar{Y}_k}{\sigma_k} \right|,$$

where, $1 \leq k \leq K$. If one of the discriminant functions in $F$ determines the class of the datum $y$ uniquely, we complete the classifying task; otherwise, if the datum is not recognized by any discriminant function or the datum is recognized by more than two discriminant functions in $F$, the $Z$-scope will be applied to determine the class to which the datum should be assigned. The detailed classification algorithm is listed as follows.

**Algorithm: The classification algorithm**

Input: An object $x$.

Output: The class $C_t$ to which $x$ is assigned.

Step 1: Transform nominal attributes of $x$ into numerical attributes $y = (w_1, w_2, \ldots, w_n)$.

Step 2: Initially, $k = 1$ and a set $\Gamma = \emptyset$.

Step 3: If $f_k(y) \geq 0$, that is, the datum $y$ is recognized by $f_k$, then $\Gamma = \{f_k\} \cup \Gamma$.

Step 4: If $k < K$, then $k = k + 1$ and go to Step 3; otherwise, go to Step 5.

Step 5: Let $|\Gamma|$ be the number of functions in $\Gamma$. If $|\Gamma| = 1$, the unique class $C_t$ corresponding to the function $f_i$ in $\Gamma$ will be returned and the algorithm halts; otherwise, go to Step 6.

Step 6: If $|\Gamma| = 0$, then we set $\Gamma = F$.

Step 7: Compute $Z_k(y)$ for each $f_k \in \Gamma$.

Step 8: Let $t = \arg \min_{k} \{Z_k(y)\}$ and the class $C_t$ is returned.

**5 EXPERIMENTAL RESULTS AND DISCUSSION**

The classifier, RMGP, proposed in this paper consists of a set of discriminant functions learned by genetic programming and an ambiguity resolution mechanism. We modified GPQuick 2.1 [29] to fit the requirements of the proposed scheme and perform the experiments. The reason of using GPQuick is that it is a well-known open source and it can be easily verified by anyone.
The experimental datasets were selected from the UCI Machine Learning repository [1]. We selected 16 datasets that are diverse in their characteristics, as shown in Table 2. Three of the selected datasets in Group 1 have nominal attributes only (led7, lymph, tic-tac-toe), nine datasets in Group 2 have numerical attributes only (breast-w, glass, ionosphere, iris, pima, sonar, vehicle, waveform, and wine) and the other four datasets in Group 3 contain both nominal and numerical attributes (australian, cleve, german, heart). The size of the datasets ranges widely from 57 cases to 5000 cases including three datasets under 200 cases, five datasets between 200 and 500 cases, five datasets between 500 and 1000 cases, and three datasets above 1000 cases. Some objects with missing values in the breast-w and the cleve datasets are deleted as [31] did. The characteristics of the selected datasets are summarized in Table 2.

The parameters of GPQuick used in our experiments are shown in Table 3. Most of them refer to the parameters used in [11] for the sake of our comparisons. Only four basic arithmetic operations {+, −, ×, ÷} are defined in GPQuick. As Kishore’s report [11], they showed that the effectiveness of a discriminant function using only these four basic operations is better than

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Attributes</th>
<th>miss</th>
<th>Classes</th>
<th>Cases</th>
<th>Attributes</th>
<th>miss</th>
<th>Classes</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>led7</td>
<td>Nominal 7</td>
<td>0</td>
<td>no 10</td>
<td>3200</td>
<td>Nominal 7</td>
<td>0</td>
<td>no 10</td>
<td>3200</td>
</tr>
<tr>
<td>lymph</td>
<td>18</td>
<td>0</td>
<td>no 4</td>
<td>148</td>
<td>18</td>
<td>0</td>
<td>no 4</td>
<td>148</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>9</td>
<td>0</td>
<td>no 2</td>
<td>958</td>
<td>9</td>
<td>0</td>
<td>no 2</td>
<td>958</td>
</tr>
<tr>
<td>breast-w</td>
<td>0</td>
<td>10</td>
<td>yes 2</td>
<td>699</td>
<td>0</td>
<td>10</td>
<td>yes 2</td>
<td>683</td>
</tr>
<tr>
<td>glass</td>
<td>0</td>
<td>9</td>
<td>no 7</td>
<td>214</td>
<td>0</td>
<td>9</td>
<td>no 7</td>
<td>214</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0</td>
<td>34</td>
<td>no 2</td>
<td>351</td>
<td>0</td>
<td>34</td>
<td>no 2</td>
<td>351</td>
</tr>
<tr>
<td>iris</td>
<td>0</td>
<td>4</td>
<td>no 3</td>
<td>150</td>
<td>0</td>
<td>4</td>
<td>no 3</td>
<td>150</td>
</tr>
<tr>
<td>pima</td>
<td>0</td>
<td>8</td>
<td>no 2</td>
<td>768</td>
<td>0</td>
<td>8</td>
<td>no 2</td>
<td>768</td>
</tr>
<tr>
<td>sonar</td>
<td>0</td>
<td>60</td>
<td>no 2</td>
<td>229</td>
<td>0</td>
<td>60</td>
<td>no 2</td>
<td>229</td>
</tr>
<tr>
<td>vehicle</td>
<td>0</td>
<td>18</td>
<td>no 4</td>
<td>846</td>
<td>0</td>
<td>18</td>
<td>no 4</td>
<td>846</td>
</tr>
<tr>
<td>waveform</td>
<td>0</td>
<td>21</td>
<td>no 3</td>
<td>5000</td>
<td>0</td>
<td>21</td>
<td>no 3</td>
<td>5000</td>
</tr>
<tr>
<td>wine</td>
<td>0</td>
<td>13</td>
<td>no 3</td>
<td>178</td>
<td>0</td>
<td>13</td>
<td>no 3</td>
<td>178</td>
</tr>
<tr>
<td>australian</td>
<td>8</td>
<td>6</td>
<td>no 2</td>
<td>690</td>
<td>8</td>
<td>6</td>
<td>no 2</td>
<td>690</td>
</tr>
<tr>
<td>cleve</td>
<td>7</td>
<td>6</td>
<td>yes 2</td>
<td>303</td>
<td>7</td>
<td>6</td>
<td>yes 2</td>
<td>296</td>
</tr>
<tr>
<td>german</td>
<td>13</td>
<td>7</td>
<td>no 2</td>
<td>1000</td>
<td>13</td>
<td>7</td>
<td>no 2</td>
<td>1000</td>
</tr>
<tr>
<td>heart</td>
<td>7</td>
<td>6</td>
<td>no 2</td>
<td>270</td>
<td>7</td>
<td>6</td>
<td>no 2</td>
<td>270</td>
</tr>
</tbody>
</table>

TABLE 2
The summary of datasets
TABLE 3
The parameters of GPQuick used in the experiments

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node mutate weight</td>
<td>43.5%</td>
<td>Crossover weight</td>
<td>28%</td>
</tr>
<tr>
<td>Mutate constant weight</td>
<td>43.5%</td>
<td>Crossover weight annealing</td>
<td>20%</td>
</tr>
<tr>
<td>Mutate shrink weight</td>
<td>13%</td>
<td>Crossover rate</td>
<td>70%</td>
</tr>
<tr>
<td>Selection method</td>
<td>Tournament</td>
<td>Population size</td>
<td>1000</td>
</tr>
<tr>
<td>Tournament size</td>
<td>7</td>
<td>Copy weight</td>
<td>4%</td>
</tr>
<tr>
<td>Mutation weight</td>
<td>8%</td>
<td>a</td>
<td>0</td>
</tr>
<tr>
<td>Mutation weight annealing</td>
<td>40%</td>
<td>p</td>
<td>10</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>10%</td>
<td>q</td>
<td>-10</td>
</tr>
<tr>
<td>Set of operations</td>
<td>{+, −, ×, ÷}</td>
<td>Generations</td>
<td>5000</td>
</tr>
</tbody>
</table>

using more operations like log, sin and logical operations. The population size is set to be 1000 and the number of maximum generations of evolution is 5000, the processing speed of evolution in GPQuick is quite fast. The reason is that GPQuick uses the steady state model. This strategy only generates one new individual and replaces one old individual for each generation at a time, as opposed to the traditional strategy that replaces all individuals in one generation (the general model). The parameters of the fitness function \( p, q \) and \( \alpha \) are set to be 10, \(-10\) and 0, respectively, for all datasets. The values of \( p, q, \) and \( \alpha \) empirically will not influence classification accuracy so much if the number of evolving generations is large enough. The main consideration for choosing the values \( p = 10, q = -10, \) and \( \alpha = 0 \) is that these values are faster than other settings when generating the discriminant functions for most of the datasets. All datasets were tested by the same parameters.

The experiments are done and estimated by the average classification error rates of 10-fold cross validation. A dataset is randomly partitioned into 10 disjoint subsets, each of approximately equal size. Training and testing are performed in 10 iterations. Each subset is used once in the test set and the other nine times in the training set. The error rate estimation is defined as

\[
\text{Error} = \frac{1}{m} \sum_{k=1}^{10} n_k
\]

where \( m \) is the number of cases in the dataset and \( n_k \) is the number of incorrect classification cases in the \( k \)th iteration. Due to the randomness of genetic programming, we performed 10 runs of 10-fold cross validation for RMGP. A new random 10-fold data set is generated in each run and the overall average error rate is the average of the 10-fold error rates for 10 runs.
The experimental results are shown in Table 4. The results are also compared with different classification models including statistical models Naive Bayes [5], NBTree [13], SNNB [31], the decision tree based classifier C4.5 [27], and the association rule-based classifier CBA [19]. These selected models had been reported that they have good effectiveness in the classification of categorical data. Although the related error rates in Table 4 are cited from [31] except RMGP, the characteristics of the test datasets are the same. Since RMGP is random-based and 10 runs performed, their standard deviations are also shown in the table.

Group 1 consists of the three datasets with nominal attributes only. RMGP is the best in the led7 dataset. NB-based classifiers have good results in the lymph dataset. RMGP is almost as good as NB-based classifiers. However, rule-based methods obviously are not suitable for such a dataset. On the contrary, in the tic-tac-toe dataset, rule-based classifiers including C4.5 and CBA have high accuracy, since the categorical attributes in the dataset contains only three different values. However, the recognition results of NB-based classifiers are poor. Although RMGP is not as good as rule-based classifiers, it is much better than NB-based classifiers in the tic-tac-toe dataset.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>NB</th>
<th>NBTree</th>
<th>SNNB</th>
<th>C4.5</th>
<th>CBA</th>
<th>RMGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>led7</td>
<td>26.7</td>
<td>26.7</td>
<td>26.5</td>
<td>26.5</td>
<td>28.1</td>
<td>23.4</td>
</tr>
<tr>
<td>lymph</td>
<td>19.0</td>
<td>17.6</td>
<td>17.0</td>
<td>26.5</td>
<td>22.1</td>
<td>17.9</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>30.1</td>
<td>17.0</td>
<td>15.4</td>
<td>0.6</td>
<td>0.4</td>
<td>4.2</td>
</tr>
<tr>
<td>breast-w</td>
<td>2.4</td>
<td>2.6</td>
<td>3.0</td>
<td>5.0</td>
<td>3.7</td>
<td>3.1</td>
</tr>
<tr>
<td>glass</td>
<td>28.5</td>
<td>28.0</td>
<td>28.0</td>
<td>31.3</td>
<td>26.1</td>
<td>27.3</td>
</tr>
<tr>
<td>ionosphere</td>
<td>10.5</td>
<td>12.0</td>
<td>10.5</td>
<td>10.0</td>
<td>7.7</td>
<td>4.9</td>
</tr>
<tr>
<td>iris</td>
<td>5.3</td>
<td>7.3</td>
<td>5.3</td>
<td>4.7</td>
<td>5.3</td>
<td>4.2</td>
</tr>
<tr>
<td>Group 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pima</td>
<td>24.5</td>
<td>24.9</td>
<td>25.1</td>
<td>24.5</td>
<td>27.1</td>
<td>25.5</td>
</tr>
<tr>
<td>sonar</td>
<td>21.6</td>
<td>22.6</td>
<td>16.8</td>
<td>29.8</td>
<td>22.5</td>
<td>16.5</td>
</tr>
<tr>
<td>vehicle</td>
<td>40.0</td>
<td>29.5</td>
<td>28.4</td>
<td>27.4</td>
<td>31.0</td>
<td>31.4</td>
</tr>
<tr>
<td>waveform</td>
<td>19.3</td>
<td>16.1</td>
<td>17.4</td>
<td>21.9</td>
<td>20.3</td>
<td>17.7</td>
</tr>
<tr>
<td>wine</td>
<td>1.7</td>
<td>2.8</td>
<td>1.7</td>
<td>7.3</td>
<td>5.0</td>
<td>5.3</td>
</tr>
<tr>
<td>Group 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>australian</td>
<td>14.1</td>
<td>14.5</td>
<td>14.8</td>
<td>15.3</td>
<td>14.6</td>
<td>13.1</td>
</tr>
<tr>
<td>cleve</td>
<td>18.1</td>
<td>19.1</td>
<td>18.5</td>
<td>21.8</td>
<td>17.1</td>
<td>20.8</td>
</tr>
<tr>
<td>german</td>
<td>24.5</td>
<td>24.5</td>
<td>26.2</td>
<td>27.7</td>
<td>26.5</td>
<td>17.3</td>
</tr>
<tr>
<td>heart</td>
<td>18.1</td>
<td>17.4</td>
<td>18.9</td>
<td>19.2</td>
<td>18.1</td>
<td>14.4</td>
</tr>
</tbody>
</table>

TABLE 4
The average error rates (%) for the compared classifiers
All the attributes of datasets in Group 2 are numeric. NB-based classifiers including NB, NBTree and SNNB lead in four domains (breast-w, pima, waveform, and wine). Rule-based classifiers lead in three domains (glass, pima and vehicle). RMGP also has the lowest error rates in three domains (ionosphere, iris and sonar). The comparison of NB-based classifiers, rule-based classifiers and RMGP results in a tie. We also found that some of datasets are easily classified by all classifiers, such as the breast-w and the iris datasets. Some datasets, like glass, pima and vehicle, are difficult to classify for all classifiers. The proposed RMGP classifier is good for the ionosphere and the sonar datasets. NB-based classifiers gain an advantage over others on the waveform and wine databases. Generally, the classification accuracy for NB-based classifiers and RMGP classifiers is better than rule-based classifiers on the datasets with numerical attributes.

In the four databases of Group 3, the proposed RMGP classifier leads in 3 datasets (australian, german and heart). It is successful here by transforming nominal attributes into the rough membership values for combining genetic programming. The classification rates are greatly improved on the above three datasets.

We made an overall evaluation for the three groups: the proposed RMGP has lower error rates than CBA in 11 out of the 16 domains, and higher error rates in 5 domains. At the same time, it presents lower error rates than C4.5 Rules in 13 domains and higher error rates in 3 domains. While comparing with Naïve Bayes, the proposed method wins 12 domains and loses in 4 domains. The RMGP leads in 7 domains and others approaches show the best classification results only in three domains at most. The computation cost for genetic programming is generally higher than NB-based or rule-based classifiers. However, the proposed learning algorithm is efficient when compared with the training time in [16] that required more than one hour. We list the learning time for each dataset in Table 5. The time in the table is the overall time of learning all discriminant functions in the domain. We found that the learning time depends on the number of discriminant functions, the number

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of $f_i$</th>
<th>Cases</th>
<th>Average</th>
<th>S. D.</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>2</td>
<td>690</td>
<td>124.35</td>
<td>21.63</td>
</tr>
<tr>
<td>breast-w</td>
<td>2</td>
<td>683</td>
<td>117.03</td>
<td>4.04</td>
</tr>
<tr>
<td>cleve</td>
<td>2</td>
<td>296</td>
<td>76.84</td>
<td>3.97</td>
</tr>
<tr>
<td>german</td>
<td>2</td>
<td>1000</td>
<td>192.99</td>
<td>20.70</td>
</tr>
<tr>
<td>glass</td>
<td>7</td>
<td>214</td>
<td>306.69</td>
<td>27.60</td>
</tr>
<tr>
<td>heart</td>
<td>2</td>
<td>270</td>
<td>89.93</td>
<td>11.65</td>
</tr>
<tr>
<td>ionosphere</td>
<td>2</td>
<td>351</td>
<td>134.32</td>
<td>5.19</td>
</tr>
</tbody>
</table>
of cases in the datasets and the difficulty of classification. More training time is needed if the domain dataset is difficult to classify.

6 CONCLUSIONS

Classification is an important task in many practical applications. The supervised learning for classification using GP is a new classification technique developed recently. The main contributions of the proposed RMGP method consist of attribute transformation and conflict resolution. The transformation algorithm based on rough membership functions solved the problem of applying nominal attributes in GP to learn discriminant functions in the learning phase. The conflict resolution handled the problems of conflict and rejection in the classification phase. The proposed scheme was tested on three types of datasets: nominal, numerical and mixed. The essential results from the experiments are concluded as follows:

1. GP can learn accurate discriminant functions from complete training data with numerical attributes.
2. The proposed transformation algorithm based on rough membership can handle nominal attributes and mixed attributes to learn effective discriminant functions by GP.
3. The proposed distance-based fitness function in GP and conflict resolution based on Z-score result in a higher accurate classifier than the one using the accuracy-based fitness function and SA measure in [11].
4. The performance of the RMGP method is above average for general complete data with only nominal attributes, only numerical attributes, or mixed attributes.
Traditional classifiers generally generate classification rules from nominal training data. In this paper, a new hybrid learning approach of combining rough membership and genetic programming are proposed to learn discriminant functions instead of classification rules for nominal attributes. A classifier with discriminant functions has the advantage of accuracy and portability. This research extends the ability of learning to handle nominal attributes and improves the effectiveness of resolving conflict among discriminant functions.

When handling training examples with missing values, the proposed RMGP method just removes them from datasets. The accuracy of classification may thus decrease if a big amount of data is removed. In the future, we will further enhance the robustness of the learning method using GP on coping with data with missing values.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable suggestions to improve the quality of this paper.

REFERENCES


