A Classifier Learning Scheme based on Rough Membership and Genetic Programming

(Workshop G: Workshop on Artificial Intelligence)

Been-Chian Chien* and Jui-Hsiang Yang

Institute of Information Engineering
I-Shou University
1, Section 1, Hsueh-Cheng Rd., Ta-Hsu Hsiang, Kaohsiung County,
Taiwan, 840, R.O.C.
E-mail: cbc@isu.edu.tw, m9003012@isu.edu.tw
TEL: +886-7-6577711 ext 6517
Fax: +886-7-6578944

and

Tzung-Pei Hong

Department of Electrical Engineering
National University of Kaohsiung
Kaohsiung, Taiwan, R.O.C.
E-mail: tphong@nuk.edu.tw

* To whom all correspondence should be sent

A Classifier Learning Scheme based on Rough Membership and Genetic Programming

Abstract

Classification is one of the important research topics in knowledge discovery and machine learning. A new classifier learning method using genetic programming has been developed for classifying numerical data recently. However, it is difficult for a function-based classifier to classify general nominal data, because nominal data may have possible large distinct values with no ordering values. In this paper, we present a new scheme based on rough set theory and genetic programming to learn a classifier from the data with both nominal and numerical attributes. The proposed scheme first transforms the nominal data into numerical values by rough membership functions. The classification functions then can be generated by genetic programming easily. We use several URI data sets to show the performance of the proposed scheme and make comparisons with other methods.

(Keywords: Knowledge discovery, Machine learning, Genetic programming, Classification, Rough set)

1. Introduction

Classification is one of the important tasks in machine learning. A classification problem is a supervised learning that is given a data set with pre-defined classes referred as training samples, then the classification rules, decision trees, or mathematical functions are learned from the training samples to classify future data with unknown class. Owing to the versatility of human activities and unpredictability of data, such mission is a challenge. For solving classification problem, many different methods have been proposed. Most of the previous classification methods are based on mathematical models or theories. For example, the probability-based classification methods are built on the Bayesian decision theory [7][9]. The Bayesian network is one of the important classification methods based on statistical model. Many improvements of Naïve Bayes like NBTree [9] and SNNB [18] also provide good classification results. Another well-known approach is neural network [20]. In the approach of neural network, a multi-layered network with m inputs and n outputs is trained with a given training set. We give an input vector to the network, and an n-dimensional output vector is obtained from the outputs of the network. Then the given vector is assigned to the class with the maximum output. The other type of classification approach uses the decision tree, such as ID3 and C4.5 [16]. A decision tree is a flow-chart-like tree structure, which each internal node denotes a decision on an attribute, each branch represents an outcome of the decision, and leaf nodes represent classes. Generally, a classification problem can be represented in a decision tree clearly.

Recently, some active techniques start to be applied by few researchers to develop new classifiers. As an example, CBA [12] employs data mining techniques to develop a hybrid rule-based classification approach by integrating classification rules mining with association

rules mining. The evolutionary computation is the other one interesting technique. The most common techniques of evolutionary computing are the genetic algorithm(GA) and the genetic programming(GP) [5][8]. For solving a classification problem, the genetic algorithm first encodes a random set of classification rules to a sequence of bit strings. Then the bit strings will be replaced by new bit strings after applying the evolution operators such as reproduction, crossover and mutation. After a number of evolving generations, the bit strings with good fitness will be generated. Thus a set of effective classification rules can be obtained from the final set of bit strings satisfying the fitness function. For the genetic programming, the classifier can be accomplished in either two ways: classification rules [5] or classification functions [8]. The main advantage of classifying by functions instead of rules is concise and efficient, because computation of functions is easier than rules induction.

The technique of genetic programming (GP) was proposed by Koza [10][11] in 1987. The genetic programming has been applied to several applications like symbolic regression, the robot control programs, and classification, etc. Genetic programming can discover underlying data relationships and presents these relationships by expressions. An expression is constructed by terminals and functions. There are several types of functions can be applied to the genetic programming:

1. Arithmetic operations: addition, subtraction, multiplication and division.
2. Trigonometric functions: Sine and Cosine, etc.
3. Conditional operators and Boolean operators: IF, ELSE and OR, etc.
4. Other add-on operations: Absolution, negative and other user-specific functions.

The algorithm of a genetic programming begins with a population that is a set of randomly created individuals. Each individual represents a potential solution that is

represented as a binary tree. Each binary tree is constructed by all possible compositions of the sets of functions and terminals. A fitness value of each tree is calculated by a suitable fitness function. According to the fitness value, a set of individuals having better fitness will be selected. These individuals are used to generate new population in next generation with genetic operators. Genetic operators generally also include reproduction, crossover, mutation and others that are used to evolve functional expressions. After the evolution of a number of generations, we can obtain an individual with good fitness value. If the fitness value of such individual still does not satisfy the specified conditions of the solution, the process of evolution will be repeated until the specified conditions are satisfied.

The previous researches on classification using genetic programming have shown the feasibility of learning classification functions by designing an accuracy-based fitness function [8] and special evolution operations[2]. However, there are two main disadvantages in the previous work. First, only numerical attributes are allowed in calculating of functions. It is difficult for genetic programming to handle the cases with nominal attributes containing categorical data. The second drawback is that classification functions may conflict each other. In this paper, we propose a new learning scheme that defines a rough attribute membership function to solve the problems of nominal attributes and gives a distance-based fitness function for genetic programming to generate a function-based classifier. Based on the distance-based fitness function, an effective conflict resolution method is developed to overcome the problem of conflicts. More than twenty datasets are selected from UCI data repository to show the performance of the proposed scheme. We also compare the results with other approaches including the statistical model, the decision tree and the association mining.

This paper is organized as follows: Section 2 introduces the concepts of rough set theory and rough membership functions. In Section 3, we discuss the proposed learning algorithm based on rough attribute membership and genetic programming. In Section 4, we present the conflicts resolution in the classification algorithm. Section 5 shows the experimental results. Finally, conclusions are made in Section 6.

2. Rough Membership Functions

Rough sets introduced by Pawlak [14] is a powerful tool for the identification of common attributes in data sets. The mathematical foundation of rough set theory is based on the set approximation of partition space on sets. The rough sets theory has been successfully applied to knowledge discovery in databases. This theory provides a powerful foundation to reveal and discover important structures in data and to classify complex objects. An attribute-oriented rough sets technique can reduce the computational complexity of learning processes and eliminate the unimportant or irrelevant attributes so that the knowledge can be learned from large databases efficiently.

The idea of rough sets is based on the establishment of equivalence classes on the given data set S and supports two approximations called lower approximation and upper approximation. The lower approximation of a concept X contains the equivalence classes that are certain to belong to X without ambiguity. The upper approximation of a concept X contains the equivalence classes that cannot be described as not belonging to X.

A vague concept description can contain boundary-line objects from the universe, which cannot be with absolute certainty classified as satisfying the description of a concept. Such uncertainty is related to the idea of membership of an element to a concept X. We use the

following definitions to describe the membership of a concept X on a specified set of attributes B [14].

Definition 1: Let $U = (S, A)$ be an information system where S is a non-empty, finite set of objects 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

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

If $(x, x') \in E_A(B)$, we say that objects x and x' are indiscernible.

Definition 2: $apr = (S, E)$, is called an approximation space. The object $x \in S$ belongs to one and only one equivalence class. Let

$$[x]_B = \{ y \mid x \, E_A(B) \, y, \forall x, y \in S \},$$

$$[S]_B = \{[x]_B \mid x \in S \}.$$

The notation $[x]_B$ denotes equivalence classes of $E_A(B)$ and $[S]_B$ denotes the set of all equivalence classes $[x]_B$ for $x \in S$.

Definition 3: For a given concept $X \subseteq S$, a rough attribute membership function of X on the set of attributes B is defined as

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

$|[x]_B|$ denotes the cardinality of equivalence classes of $[x]_B$. $|[x]_B \cap X|$ denotes the cardinality of the set $[x]_B \cap X$. The rough membership value $\mu_B^X(x)$ can be interpreted as the

conditional probability that an object x belongs to X, given that the object belongs to $[x]_B$. The value of $\mu_B^X(x)$ is in the range of [0, 1].

3. The Learning Algorithm of Classification Functions

3.1 Classification Functions

Consider a given data set S, for a data x_j such that $x_j \in S$ having n attributes $A_1, A_2, \ldots, A_n$. Let $A = \{A_1, A_2, \ldots, A_n\}$ and $A_i \in \mathbf{R}$, for $1 \le i \le n$. Assume that

$$x_j = (v_{j1}, v_{j2}, \ldots, v_{jn}),$$

where $v_{jt} \in A_t$ stands for the t-th attribute of data x_j in S. Let $C = \{C_1, C_2, \ldots, C_K\}$ be the set of K predefined classes. We may say that $<x_j, c_j>$ is a sample if the data x_j belongs to class c_j, $c_j \in C$. We define a training set (TS) to be a set of samples,

$$TS = \{<x_j, c_j> \mid x_j \in S, c_j \in C, 1 \le j \le m\}.$$

Where $m=|TS|$ is the number of samples in TS, and m_i is the number of samples belonging to the class C_i,

$$m = \sum_{i=1}^{K} m_i, \quad 1 \le i \le K.$$

A classification function for class C_i, f_i, is a function

$$f_i : \mathrm{R}^n \to \mathrm{R},$$

such that satisfies the following conditions:

$$\begin{cases} f_i(x_j) \ge a, \text{if} & c_j = C_i \\ f_i(x_j) < a, \text{if} & c_j \ne C_i \end{cases}, \text{ where } 1 \le i \le K, 1 \le j \le m.$$

A set of classification functions F for the set of class C is defined as

$$F = \{f_i \mid f_i : \mathrm{R}^n \to \mathrm{R}, 1 \le i \le K\}.$$

3.2 The Transformation of Rough Attributes Membership

The classification function defined in Section 3.1 has a limitation on attributes. Since the calculation of functions allows only numerical values, it cannot work if dataset contained nominal attributes. In order to apply the genetic programming to train the data set, we make use of rough attribute membership as the definitions in Section 2 to transform the nominal attributes into a set of numerical attributes.

For the set of n attributes $A = \{A_1, A_2, ..., A_n\}$, a data $x_j \in S$, $x_j = (v_{j1}, v_{j2}, ... , v_{jn})$, $v_{ji} \in A_i$. If A_i is a numerical attribute, we have $\tilde{A}_i = \{A_i\}$, let w_{jk} be the value of $\tilde{A}_i$, $w_{jk} = v_{ji}$.

If A_i is a nominal attribute, we assume that S is partitioned into p_i equivalence classes by attribute A_i. Let $[s_l]_{Ai}$ denote the l different partitions on attribute A_i, p_i is the number of partitions on the attribute A_i. Thus, we have

$$[S]_{Ai} = [S]_{A_i} = \bigcup_{l=1}^{p_i} [s_l]_{A_i} \text{ , where } p_i = |[S]_{Ai}|.$$

We transform the original nominal attribute A_i into a set of K numerical attributes $\tilde{A}_i$. Let

$$\tilde{A}_i = \{A_{i1}, A_{i2}, ... , A_{iK}\},$$

where K is the number of predefined classes C as defined in Section 3.1. Let the values of $\tilde{A}_i$ be denoted as

$$(w_{jk}, w_{j(k+1)}, ... , w_{j(k+K-1)}), w_{ik} \in A_{ik}.$$

For a data $x_j \in S$, $x_j = (v_{j1}, v_{j2}, ... , v_{jn})$, $v_{ji} \in A_i$ is a nominal attribute, we have

$$w_{jk} = \mu_{A_i}^{C_1}(x_j), w_{j(k+1)} = \mu_{A_i}^{C_2}(x_j), ... , w_{j(k+K-1)} = \mu_{A_i}^{C_K}(x_j),$$

where

$$\mu_{A_i}^{C_k}(x_j) = \frac{|[s_l]_{A_j} \cap [x_j]_{C_k}|}{|[s_l]_{A_j}|}, \text{ if } v_{ji} \in [s_l]_{Ai}.$$

After the transformation, we get the new set of attributes $\tilde{A}$ and the value y_j, as follows

$$\tilde{A} = \bigcup_{i=1}^{n} \tilde{A}_i \ , \ y_j = (w_{j1}, w_{j2}, \ldots, w_{jn'}),$$

where $n' = (n\text{-}r) + rK$, r is the number of nominal attributes in A. Thus, the new training set becomes

$$TS' = \{<y_j, c_j> | \ x_j \in S \ , c_j \in C, 1 \le j \le m\}.$$

3.3 The Fitness Function

The fitness value is important for genetic programming to evaluate an individual and generate effective solutions. From the definition of claiming classification functions in Section 3.1, we consider a classification function f_i of a class C_i and a specified constant a. For the positive instances $<y_j, c_j>$, $c_j = C_i$ in the training set TS', we urge that $f_i(y_j) \ge a$; on the contrary, $f_i(y_j) < a$ for negative instance $<y_j, c_j>$, $c_j \ne C_i$. To achieve the objective of f_i, we define two parameters p and q, let $p > a$, $q < a$ and $p + q = 2 \cdot a$. We measure the error of a positive instance by

$$D_p = \begin{cases} 0 & if \quad c_j = C_i \quad and \quad f_i(y_j) \ge a \\ [p - f_i(y_j)]^2 & if \quad c_j = C_i \quad and \quad f_i(y_j) < a \end{cases},$$

and measure the error of a negative instance by

$$D_n = \begin{cases} 0 & if \quad c_j \ne C_i \quad and \quad f_i(y_j) < a \\ [f_i(y_j) - q]^2 & if \quad c_j \ne C_i \quad and \quad f_i(y_j) \ge a \end{cases}.$$

The fitness value of an individual is then evaluated by the following fitness function:

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

where m is the number of training samples, $<y_j, c_j> \in TS'$, $1 \le j \le m$. Since the fitness value of an individual represents the degree of error between target function and the individual, we have the fitness value be as large as possible and approach to zero.

3.4 The Learning Algorithm

The learning algorithm for classification functions using genetic programming is described in detail as follow:

Algorithm: The genetic programming for learning classification functions

Input: The training set TS

Output: A function with the best fitness value

Step 1: Initial value $i = 1$, $k = 1$.

Step 2: Transform nominal attributes into rough attribute membership values.

For a data $x_j \in TS$, $x_j = (v_{j1}, v_{j2}, \ldots, v_{jn})$, for all $1 \le j \le m$,

If A_i is a numerical attribute, $w_{jk} = v_{ji}$, $k = k + 1$.

If A_i is a nominal attribute, $w_{jk} = \mu_{A_i}^{C_1}(x_j)$, $w_{j(k+1)} = \mu_{A_i}^{C_2}(x_j)$, ... , $w_{j(k+K-1)} = \mu_{A_i}^{C_K}(x_j)$,

$k = k + K$, repeat Step 2 until $y_j = (w_{j1}, w_{j2}, \ldots, w_{jn'})$ is generated, $n' = (n\text{-}r) + rK$, r is the number of nominal attributes in A.

Step 3: The new training set $TS' = \{<y_j, c_j> | y_j = (w_{j1}, w_{j2}, \ldots, w_{jn'}), c_j \in C, 1 \le j \le m\}$.

Step 4: Initialize the population.

Let *gen* = 1 and generate the set of individuals $\mathbf{\Omega}_1 = \{ h_1^1, h_2^1, \ldots, h_q^1 \}$ initially, where $\mathbf{\Omega}_{(gen)}$ is the population in the generation *gen* and $h_i^{(gen)}$ stands for the *i*th individual of the generation *gen*.

Step 5: Evaluate the fitness value of each individual on the training set.

For all $h_i^{(gen)} \in \mathbf{\Omega}_{(gen)}$, compute the fitness values $E_i^{(gen)} = fitness(h_i^{(gen)}, TS')$, where the fitness evaluating function *fitness*() is dependent on the problem and the function is defined by the user.

Step 6: Does it satisfy the conditions of termination?

If the best fitness value of $E_i^{(gen)}$ satisfies the conditions of termination or the *gen* is equal to the specified maximum generation, then the $h_i^{(gen)}$ with the best fitness value is returned and the algorithm halts; otherwise, *gen* = *gen* + 1.

Step 7: Generate the next generation of individuals and go to Step 5.

The new population of next generation $\mathbf{\Omega}_{(gen)}$ is generated by the ratio of P_r, P_c and P_m, goes to Step 5, where P_r, P_c and P_m represent the probabilities of reproduction, crossover and mutation operations, respectively.

3.5 An Example

We give an example to explain the above learning algorithm more clearly. The example used in Table 1 is the weather set. This data set concerns the conditions if it is suitable for playing some unspecified game. The conditions consist of four nominal attributes: *outlook*, *temperature*, *humidity*, and *wind*. The outcome is whether to play or not. The symbolic categories in the four attributes, respectively, are:

Table 1. The weather data set.

data	*outlook*	*temperature*	*humidity*	*windy*	*play*
x_1	sunny	hot	high	false	no
x_2	sunny	hot	high	true	no
x_3	overcast	hot	high	false	yes
x_4	rainy	mild	high	false	yes
x_5	rainy	cool	normal	false	yes
x_6	rainy	cool	normal	true	no
x_7	overcast	cool	normal	true	yes
x_8	sunny	mild	high	false	no
x_9	sunny	cool	normal	false	yes
x_{10}	rainy	mild	normal	false	yes
x_{11}	sunny	mild	normal	true	yes
x_{12}	overcast	mild	high	true	yes
x_{13}	overcast	hot	normal	false	yes
x_{14}	rainy	mild	high	true	no

■ *outlook*: {sunny, overcast, rainy},

■ *temperature*: {hot, mild, cool},

■ *humidity*: {high normal},

■ *windy*: {true, false}.

We give an example for showing the learning algorithm for the class *play* = yes.

Step 1: Initial value $i = 1$, $k = 1$.

Step 2: Transform nominal attributes into rough attribute membership values.

Let $A_1 = \{outlook\}$, $[s_1]_{A1}$ be the partition of *outlook* = sunny, $[s_2]_{A1}$ be the partition of *outlook* = overcast, and $[s_3]_{A1}$ be the partition of *outlook* = *rainy*.

$[s_1]_{A1} = [x_1]_{A1} = [x_2]_{A1} = [x_8]_{A1} = [x_9]_{A1} = [x_{11}]_{A11} = \{x_1, x_2, x_8, x_9, x_{11}\}$,

$[s_2]_{A1} = [x_3]_{A1} = [x_7]_{A1} = [x_{12}]_{A1} = [x_{13}]_{A1} = \{x_3, x_7, x_{12}, x_{13}\}$,

$[s_3]_{A1} = [x_4]_{A1} = [x_5]_{A1} = [x_6]_{A1} = [x_{10}]_{A1} = [x_{14}]_{A1} = \{x_4, x_5, x_6, x_{10}, x_{14}\}$.

$[S]_{A_1}=\{\{x_1, x_2, x_8, x_9, x_{11}\}, \{x_3, x_7, x_{12}, x_{13}\}, \{x_4, x_5, x_6, x_{10}, x_{14}\}\}$, $p_1 = |[S]_{A_1}| = 3$.

Let A_2 = {*temperature*}, $[s_1]_{A_2}$ be the partition of *temperature* = hot, $[s_2]_{A_2}$ be the partition of *temperature* = mild, and $[s_3]_{A_2}$ be the partition of *temperature* = cool.

$[s_1]_{A_2} = [x_1]_{A_2} = [x_2]_{A_2} = [x_3]_{A_2} = [x_{13}]_{A_2} = \{x_1, x_2, x_3, x_{13}\}$,

$[s_2]_{A_2} = [x_4]_{A_2} = [x_8]_{A_2} = [x_{10}]_{A_2} = [x_{11}]_{A_2} = [x_{12}]_{A_2} = [x_{14}]_{A_2} = \{x_4, x_8, x_{10}, x_{11}, x_{12}, x_{14}\}$,

$[s_3]_{A_2} = [x_5]_{A_{23}} = [x_6]_{A_{23}} = [x_7]_{A_{23}} = [x_9]_{A_{23}} = \{x_5, x_6, x_7, x_9\}$.

$[S]_{A_2}=\{\{x_1, x_2, x_3, x_{13}\}, \{x_4, x_8, x_{10}, x_{11}, x_{12}, x_{14}\}, \{x_5, x_6, x_7, x_9\}\}$, $p_2 = |[S]_{A_1}| = 3$.

Let A_3 = {*humidity*}, $[s_1]_{A_3}$ be the partition of *humidity* = high and $[s_2]_{A_3}$ be the partition of *humidity* = normal.

$[s_1]_{A_3} = [x_1]_{A_3} = [x_2]_{A_3} = [x_3]_{A_3} = [x_4]_{A_3} = [x_8]_{A_3} = [x_{12}]_{A_3} = [x_{14}]_{A_3} = \{x_1, x_2, x_3, x_4, x_8, x_{12}, x_{14}\}$,

$[s_2]_{A_3} = [x_5]_{A_3} = [x_6]_{A_3} = [x_7]_{A_3} = [x_9]_{A_3} = [x_{10}]_{A_3} = [x_{11}]_{A_3} = [x_{13}]_{A_3} = \{x_5, x_6, x_7, x_9, x_{10}, x_{11}, x_{13}\}$.

$[S]_{A_3}=\{\{x_1, x_2, x_3, x_4, x_8, x_{12}, x_{14}\}, \{x_5, x_6, x_7, x_9, x_{10}, x_{11}, x_{13}\}\}$, $p_3 = |[S]_{A_3}| = 2$.

Let A_4 = {*windy*}, $[s_1]_{A_4}$ be the partition of *windy* = true, $[s_2]_{A_4}$ be the partition of *windy* = false.

$[s_1]_{A_4} = [x_2]_{A_4} = [x_6]_{A_4} = [x_7]_{A_4} = [x_{11}]_{A_4} = [x_{12}]_{A_4} = [x_{14}]_{A_4} = \{x_2, x_6, x_7, x_{11}, x_{12}, x_{14}\}$.

$[s_1]_{A_4} = [x_1]_{A_4} = [x_3]_{A_4} = [x_4]_{A_4} = [x_5]_{A_4} = [x_8]_{A_4} = [x_9]_{A_4} = [x_{10}]_{A_4} = [x_{13}]_{A_4}$

$= \{x_1, x_3, x_4, x_5, x_8, x_9, x_{10}, x_{13}\}$.

$[S]_{A_4}=\{\{x_2, x_6, x_7, x_{11}, x_{12}, x_{14}\}, \{x_1, x_3, x_4, x_5, x_8, x_9, x_{10}, x_{13}\}\}$, $p_4 = |[S]_{A_4}| = 2$.

For C_1: *play* = yes,

$[x_3]_{C_1} = [x_4]_{C_1} = [x_5]_{C_1} = [x_7]_{C_1} = [x_9]_{C_1} = [x_{10}]_{C_1} = [x_{11}]_{C_1} = [x_{12}]_{C_1} = [x_{13}]_{C_1}$

$= \{x_3, x_4, x_5, x_7, x_9, x_{10}, x_{11}, x_{12}, x_{13}\}$.

For C_2: *play* = no,

$[x_1]_{C_2} = [x_2]_{C_2} = [x_6]_{C_2} = [x_8]_{C_2} = [x_{14}]_{C_2} = \{x_1, x_2, x_6, x_8, x_{14}\}$.

Table 2. The rough attribute membership values of the weather data.

Data	*outlook*		*temperature*		*humidity*		*wind*		*play*
	w_{j1}	w_{j2}	w_{j3}	w_{j4}	w_{j5}	w_{j6}	w_{j7}	w_{j8}	C_i
y_1	0.40	0.60	0.50	0.50	0.43	0.57	0.75	0.25	no
y_2	0.40	0.60	0.50	0.50	0.43	0.57	0.50	0.50	no
y_3	1.00	0.00	0.50	0.50	0.43	0.57	0.75	0.25	yes
y_4	0.60	0.40	0.67	0.33	0.43	0.57	0.75	0.25	yes
y_5	0.60	0.40	0.75	0.25	0.86	0.14	0.75	0.25	yes
y_6	0.60	0.40	0.75	0.25	0.86	0.14	0.50	0.50	no
y_7	1.00	0.00	0.75	0.25	0.86	0.14	0.50	0.50	yes
y_8	0.40	0.60	0.67	0.33	0.43	0.57	0.75	0.25	no
y_9	0.40	0.60	0.75	0.25	0.86	0.14	0.75	0.25	yes
y_{10}	0.60	0.40	0.67	0.33	0.86	0.14	0.75	0.25	yes
y_{11}	0.40	0.60	0.67	0.33	0.86	0.14	0.50	0.50	yes
y_{12}	1.00	0.00	0.67	0.33	0.43	0.57	0.50	0.50	yes
y_{13}	1.00	0.00	0.50	0.50	0.86	0.14	0.75	0.25	yes
y_{14}	0.60	0.40	0.67	0.33	0.43	0.57	0.50	0.50	no

The number of nominal attributes $n' = 4\times2 = 8$.

$$w_{11} = \mu_{A_1}^{C_1}(x_1) = 0.4\,,\ w_{12} = \mu_{A_1}^{C_2}(x_1) = 0.6\,,$$

$$w_{13} = \mu_{A_2}^{C_1}(x_1) = 0.5\,,\ w_{14} = \mu_{A_2}^{C_2}(x_1) = 0.5\,,$$

$$w_{15} = \mu_{A_3}^{C_1}(x_1) = 0.43\,,\ w_{16} = \mu_{A_3}^{C_2}(x_1) = 0.57\,,$$

$$w_{17} = \mu_{A_4}^{C_1}(x_1) = 0.75\,,\ w_{18} = \mu_{A_4}^{C_2}(x_1) = 0.25\,.$$

$$y_1 = (w_{11}, w_{12}, \ldots, w_{18}) = (0.4, 0.6, 0.5, 0.5, 0.43, 0.57, 0.75, 0.25).$$

The final results are listed in Table 2.

Step 3: Let $TS'=\{\langle y_j, c_j\rangle \mid y_j = (w_{j1}, w_{j2}, \ldots, w_{j8})\ , c_j \in C, 1 \le j \le 14\}$, listed as Table 2.

Step 4: Initialize the parameters.

We set m = 14, a = 0, p = 100, and q = 100 for the fitness function. Initially, *gen* = 1, we generate $\mathbf{\Omega}_1 = \{h_1^1, h_2^1, \ldots, h_q^1\}$. The set of positive instance of *TS*' for classification C_1 is $\{y_3, y_4, y_5, y_7, y_9, y_{10}, y_{11}, y_{12}, y_{13}\}$, and the set of negative instances is $\{y_1, y_2, y_6, y_8, y_{14}\}$.

Step 5: Evaluate the fitness values of each individual on the training set.

Assume that $h_1^1 = 50(w_{j1}\text{-}w_{j2})$, we compute the fitness values $E_1^1 = fitness(h_1^1, TS')$.

The fitness value of the individual h_1^1 is

$$E_1^1 = fitness(h_1^1, TS') = -\sum_{j=1}^{14}(D_p + D_n) = \text{-}40400.$$

Step 6: Does it satisfy the conditions of termination?

For above case, if the best fitness value of $E_i^{(gen)}$ does not satisfy the condition of termination, then *gen* = *gen* + 1 and go to Step 7. However, if the best fitness value of $h_i^{(gen)}$ satisfies the condition, $h_i^{(gen)}$ is returned and the algorithm halts. After the computing for a finite number of generations, we can get the solution. In this example, the final function is as follows

$$f_{C1} = 100[(\text{-}(w_{j8}\text{-}w_{j7})(w_{j3}\text{-}w_{j4})+(w_{j1}\text{-}w_{j2})+(w_{j5}\text{-}w_{j6}))\text{-}7.$$

By the same learning procedure, the classification function f_{C2} can be obtained:

$$f_{C_2} = \frac{1/w_{j4}}{-[54(w_{j7} - w_{j8}) + 100(w_{j1} - w_{j2}) + 12(w_{j3} - w_{j4}) + 3]}.$$

Step 7: Generate the next generation of individuals and go to Step 5.

The new population of next generation $\mathbf{\Omega}_{(gen)}$ is generated by the ratio of P_r, P_c and P_m and goes to Step 5.

4. The Classification Algorithm

After generating the classification functions, the task of classification becomes straight and simple calculations of mathematical formulas. However, a classifier cannot recognize all objects correctly in real applications generally. Except the case of misclassification, two situations of conflict may occur. The first case is that an object is recognized by two or more classification functions at the same time. The other case is that an object cannot be recognized by any classification function. Under the above both situations, we cannot decide the exact class while classifying an unknown object. A complete classifier will include a conflict resolution method to solve the problem of conflict usually. The resolution in the proposed scheme is based on the distance-based fitness values and *Z*-score of statistical test. We present the resolution approach in the following.

For a classification function $f_i \in F$ and samples $\langle y_j, c_j \rangle \in TS'$ with $c_j = C_i$, let $\overline{X}_i$ be the mean of values of $f_i(y_j)$, $1 \le j \le m_i$. That is,

$$\overline{X}_i = \frac{\sum_{\substack{<y_j, c_j> \in TS, \\ c_j = C_i}} f_i(y_j)}{m_i}, 1 \le j \le m_i, 1 \le i \le K.$$

For each $\overline{X}_i$, the standard deviation of values of $f_i(y_j)$, $1 \le j \le m_i$, is defined as

$$\sigma_i = \sqrt{\frac{\sum_{\substack{<y_j, c_j> \in TS, \\ c_j = C_i}} (f_i(y_j) - \overline{X}_i)^2}{m_i}}, 1 \le j \le m_i, 1 \le i \le K.$$

For a data $x \in S$ and a classification function f_i, let $y \in S'$ be the data with all numerical values after transforming x using rough attribute membership. The *Z*-value of data y for f_i is defined as

$$Z_i(y_j) = \frac{|f_i(y_j) - \overline{X}_i|}{\sigma_i / \sqrt{m_i}},$$

where, $1 \le i \le K$. If one of the classification functions in F determines the class of the data y uniquely, we complete the classifying task. However, once the data cannot be recognized by any classification function or the data is recognized by more than two classification functions in F, the Z-value will be applied to determine the class to which the data should be assigned. The detailed classification algorithm is listed as follow.

Algorithm: The classification algorithm

Input: A data x

Output: The class C_k that x is assigned

Step 1: Initial value $k = 1$.

Step 2: Transform nominal attributes of x into numerical attributes.

Assume that the data $x \in S$, $x = (v_1, v_2, \ldots, v_n)$.

If A_i is a numerical attribute, $w_k = v_i$, $k = k + 1$.

If A_i is a nominal attribute, $w_k = \mu_{A_i}^{C_1}(x)$, $w_{j(k+1)} = \mu_{A_i}^{C_2}(x)$, ... , $w_{j(k+K-1)} = \mu_{A_i}^{C_K}(x)$,

$k = k + K$, repeat Step 2 until $y = (w_1, w_2, \ldots, w_{n'})$ is generated, $n' = (n\text{-}r) + rK$, r is the number of nominal attributes in A.

Step 3: Initially, $i = 1$ and there exists a set $\boldsymbol{Z}$ such that $\boldsymbol{Z} = \varnothing$.

Step 4: If $f_i(y) \ge 0$, that is, the data y is recognized by f_i, then $\boldsymbol{Z} = \{f_i\} \cup \boldsymbol{Z}$.

Step 5: If $i < K$, then $i = i + 1$, go to Step 4. Otherwise, go to Step 6.

Step 6: Let $|\boldsymbol{Z}|$ be the number of functions in $\boldsymbol{Z}$. If $|\boldsymbol{Z}| = 1$, the unique class C_i corresponding to the function f_i in $\boldsymbol{Z}$ will be returned and stop; otherwise, go to Step 7.

Step 7: If $|\boldsymbol{Z}| = 0$, $\boldsymbol{Z} = F$.

Step 8: Compute $Z_i(y)$, where $f_i \in \boldsymbol{Z}$.

Step 9: Find the $k = \arg\min_{i} \{Z_i(y)\}$, the data x will be assigned to the class C_k.

5. The Experimental Results

The proposed learning algorithm based on genetic programming is implemented by modifying the GPQuick 2.1 [17]. The parameters used in our experiments are list in Table 3. We define only four basic operations {+, -, ×, ÷} for final functions. That is, each classification function contains only the four basic operations. The experimental data sets are selected from UCI Machine Learning repository [1]. We take 20 data sets from the repository totally including 3 nominal data sets, 7 composite data sets (with nominal and numeric attributes), and 10 numerical data sets. For avoiding the missing values in attributes, we modified some of data sets by deleting some objects and attributes with incomplete data into our experimental data sets. The related information of the selected data sets is summarized in Table 4.

Table 3. The parameters of GPQuick used in the experiments.

Parameter	Value
Node mutate weight	43.5%
Mutate constant weight	43.5%
Mutate shrink weight	13%
Selection method	Tournament
Tournament size	7
Crossover weight	28%
Crossover weight annealing	20%
Mutation weight	8%
Mutation weight annealing	40%
Population size	1000
Set of functions	{+, -, ×, ÷}
Generations	5000

Table 4. The information of data sets

Dataset	Original Characteristics					Modified Characteristics			
	Attributes		Miss	Classes	Cases	Attributes		Classes	Cases
	nominal	numeric				nominal	numeric		
australian	8	6	no	2	690	8	6	2	690
breast-w	0	10	yes	2	699	0	10	2	683
cleve	7	6	yes	2	303	7	6	2	296
diabetes	0	8	no	2	768	0	8	2	768
german	13	7	no	2	1000	13	7	2	1000
glass	0	9	no	7	214	0	9	7	214
heart	7	6	no	2	270	7	6	2	270
horse	15	7	yes	2	368	9	4	2	326
ionosphere	0	34	no	2	351	0	34	2	351
iris	0	4	no	3	150	0	4	3	150
labor	8	8	no	2	57	3	4	2	57
led7	7	0	no	10	3200	7	0	10	3200
lymph	18	0	no	4	148	18	0	4	148
pima	0	8	no	2	768	0	8	2	768
sick	22	7	yes	2	2800	22	1	2	2800
sonar	0	60	no	2	229	0	60	2	229
tic-tac-toe	9	0	no	2	958	9	0	2	958
vehicle	0	18	no	4	846	0	18	4	846
waveform	0	21	no	3	5000	0	21	3	5000
wine	0	13	no	3	178	0	13	3	178

The performance of the proposed classification scheme is evaluated by the average classification error rate for 10 runs of 10-fold cross validation. We figure out the experimental results and compare the effectiveness with different classification models in Table 5. These models include statistical model like Naïve Bayes [4], NBTree[9], SNNB [18]; decision tree based like C4.5 [16], and the association rule-based classifier CBA [12]. Since the proposed GP-based classifier is random based, we also show the standard deviations in the table for reference of readers.

From the experimental results, we observed that the proposed method obtains lower error rates than CBA in 11 out of the 20 domains, and higher error rates in 9 domains. It obtains lower error rates than C4.5 Rules in 14 domains and higher error rates 6 domains. While comparing our method with NBTree and SNNB, the result is in a tie. While comparing with Naïve Bayes, the proposed method wins 12 domains and loses in 8 domains. Generally, the classification results of proposed method is better than others on an average. However, in some data sets, the test results in GP-based is much worse than others, for example, in the "labor" data, we found that the average error rate is 23.1. The main reason of high error rate terribly in this case is the small size of samples in the data set. The "labor" contains only 57 data totally and is divided into two classes. While the data with small size is tested in 10-fold cross validation, the situation of overfit will occur between the two classification functions easily. The other reason is that we delete some of the attributes for avoiding the problem of missing values in data. Although the data may have missing values in only one or two attributes of some objects, deleting entire the objects or attributes straight will decrease the accuracy of classification. The results of data sets "horse" and "sick" have the same effect as "labor". Nevertheless, the proposed GP-based classification scheme still have good results in general.

Table 5. Experimental results and comparisons.

Dataset	NB	NBTree	SNNB	C4.5	CBA	GP-Based	
	[18]	[18]	[18]	[18]	[18]	Average	S.D.
australian	14.1	14.5	14.8	15.3	14.6	**13.1**	0.6
breast-w	2.4	2.6	3.0	5.0	3.7	3.1	0.4
cleve	18.1	19.1	18.5	21.8	17.1	20.8	1.3
diabetes	24.1	24.1	24.1	25.8	25.5	25.5	1.7
german	24.5	24.5	26.2	27.7	26.5	**17.3**	0.8
glass	28.5	28.0	28.0	31.3	26.1	27.3	1.3
heart	18.1	17.4	18.9	19.2	18.1	**14.4**	0.4
horse	21.7	18.7	17.4	17.4	17.6	23.2	1.8
ionosphere	10.5	12.0	10.5	10.0	7.7	**4.9**	0.5
iris	5.3	7.3	5.3	4.7	5.3	**4.2**	0.8
labor	5.0	12.3	3.3	20.7	13.7	23.1	1.9
led7	26.7	26.7	26.5	26.5	28.1	**23.4**	2.1
lymph	19.0	17.6	17.0	26.5	22.1	17.9	1.6
pima	24.5	24.9	25.1	24.5	27.1	25.5	2.0
sick	4.2	22.1	3.8	1.5	2.8	11.7	0.7
sonar	21.6	22.6	16.8	29.8	22.5	**16.5**	0.9
tic-tac-toe	30.1	17.0	15.4	0.6	0.4	4.2	0.4
vehicle	40.0	29.5	28.4	27.4	31	31.4	2.2
waveform	19.3	16.1	17.4	21.9	20.3	17.7	1.5
wine	1.7	2.8	1.7	7.3	5.0	5.3	0.4

6. Conclusions

Classification is an important task in many applications. The technique of classification using genetic programming is a new classification approach developed recently. However, how to handling nominal attributes in genetic programming is a difficult problem. We proposed a scheme based on the rough membership function to classify data with nominal attribute using genetic programming in this paper. Furthermore, we give a conflict resolution

mechanism for avoiding conflicting among classification functions. The experimental results demonstrate that the proposed scheme is feasible. Although it is not so effective in some cases of datasets, we are trying to find a new fitness function to improve the accuracy for any possible datasets and cope with the data having missing values in the future.

References

[1] C. Blake, E. Keogh, C. J. Merz, UCI repository of machine learning database, http://www.ics.uci.edu/~mlearn/MLReopsitory.html, Irvine, University of California, Department of Information and Computer Science, 1998.

[2] M. Bramrier and W. Banzhaf, A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining, *IEEE Transaction on Evolutionary Computation*, Vol. 5, No. 1, Feb. pp. 17-26, 2001.

[3] B. C. Chien, J. Y. Lin, and T. P. Hong, Learning Discriminant Functions with Fuzzy Attributes for Classification Using Genetic Programming, *Expert Systems with Applications*, No. 23, pp. 31-37, 2002.

[4] R. O. Duda, P. E. Hart, Pattern Classification and Scene Analysis, *New York: John Wiley*, 1973.

[5] A. A. Freitas, A Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction, *Proceedings of the 2nd Annual Conference Morgan Kaufmann*, pp. 96-101, 1997.

[6] E. H. Han, G. Karypis, V. Kumar, Text Categorization Using Weight Adjusted *k*-nearest Neighbor Classification, *PhD thesis, University of Minnesota*, 1999.

[7] D. Heckerman, M. P. Wellman, “Bayesian networks”, *Communications of the ACM*, Vol.

38, No. 3, pp. 27-30, 1995.

[8] J. K. Kishore, L. M. Patnaik, V. K. Agrawal, Application of Genetic Programming for Multicategory Pattern Classification, *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 242-258, 2000.

[9] R. Kohavi, Scaling Up the Accuracy of Naïve-Bayes Classifiers: a Decision-Tree Hybrid. *Proceedings of the Second International Conference on Knowledge Discovery & Data Mining, AAAI Press/MIT press,* Cambridge/Menlo Park, pp. 202-207, 1996.

[10] J. R. Koza, Genetic Programming: "On the programming of computers by means of Natural Selection", *MIT Press*, 1992.

[11] J. R. Koza, Introductory Genetic Programming Tutorial, *Genetic Programming 1996 Conference, Stanford University*, 1996.

[12] B. Liu, W. Hsu, and Y. Ma, Integrating Classification and Association Rule Mining. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 443-447, 1998.

[13] T. Loveard and V. Ciesielski, Representing Classification Problems in Genetic Programming, in *Proceedings of the 2001 Congress on Evolutionary Computation*, pp. 1070-1077, 2001.

[14] Z. Pawlak, Rough Sets, *International Journal of Computer and Information Sciences*, No. 11, pp. 341-356,1982.

[15] Z. Pawlak, A. Skowron, Rough Membership Functions, in: R.R. Yager and M. Fedrizzi and J. Kacprzyk (Eds.), *Advances in the Dempster-Shafer Theory of Evidence*, pp. 251-271, 1994.

[16] J. R. Quinlan, C4.5: Programs for Machine Learning, *Morgan Kaufmann*, 1993.

[17] A. Singleton, Genetic Programming with C++, *Byte*, Feb. pp. 171-176, 1994.

[18] Z. Xie, W. Hsu, Z. Liu, M. L. Lee, SNNB: A Selective Neighborhood Based Naïve Bayes for Lazy Learning, *Proceedings of the sixth Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 104-114, 2002.

[19] Y. Y. Yao, S. K. M. Wong, A Decision Theoretic Framework for Approximating Concepts, *International Journal of Man-machine Studies*, No. 37, pp. 793-809, 1992.

[20] G. P. Zhang, Neural Networks for Classification: a Survey, *IEEE Transaction on Systems, Man, And Cybernetics-Part C: Applications and Reviews*, Vol.30, No. 4, pp. 451-462, 2000.