PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 1957–1972

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Learning effective classifiers with Z-value measure based on genetic programming

Been-Chian Chien[a,*], Jung-Yi Lin[b], Wei-Pang Yang[b]

[a]*Department of Computer Science and Information Engineering, National Tainan Teachers College, 33, Section 2, Su-Lin street, Tainan 700, Taiwan, R.O.C.*
[b]*Department of Computer and Information Science, National Chiao Tung University, Hsin-Chu, Taiwan, ROC*

Received 20 October 2003; accepted 8 March 2004

Abstract

This paper presents a learning scheme for data classification based on genetic programming. The proposed learning approach consists of an adaptive incremental learning strategy and distance-based fitness functions for generating the discriminant functions using genetic programming. To classify data using the discriminant functions effectively, the mechanism called Z-value measure is developed. Based on the Z-value measure, we give two classification algorithms to resolve ambiguity among the discriminant functions. The experiments show that the proposed approach has less training time than previous GP learning methods. The learned classifiers also have high accuracy of classification in comparison with the previous classifiers.

Keywords: Knowledge discovery; Machine learning; Genetic programming; Classification; Z-value measure

1. Introduction

Data classification is one of the important issues in the research area of machine learning. Many applications can be viewed as extensions of classification problem. For example, pattern recognition, disease diagnosis, and business decision-making. The classification is a two-step process including learning and classifying generally. In the first step, we referred to a predetermined set of data as training data set, which is used to build a classifier by a learning algorithm. The learned classifier is then used for classification. The task of classification is to assign an unknown object to one of the predefined classes based on the observed attributes of the object. Since the versatility of human activities and the unpredictability of data, it is a challenge for researchers to learn an effective classifier efficiently.

To reduce the learning time and increase the classification accuracy, many different methods for building effective classifiers have been proposed in the past decades. Typical rule-based classifiers like ID3 [1] and C4.5 [2] construct decision trees and classification rules using entropy-based measure called information gain. Some of the previous methods are based on some mathematical models or theories. For example, the statistical classifiers are built on the Bayesian decision theory [3–5]. The theory provides a probability model for classification by minimizing the probability of total misclassification rate. Another well-known approach is neural network [6–9]. In neural network approach, a multilayered network with m inputs and n outputs is trained with a given set of training data. 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

* Corresponding author. Tel.: +886-6-2133111; fax: +886-6-2144409.
E-mail address: bcchien@ipx.ntntc.edu.tw (B.-C. Chien).

0031-3203/$ - see front matter
doi:10.1016/j.patcog.2004.03.016

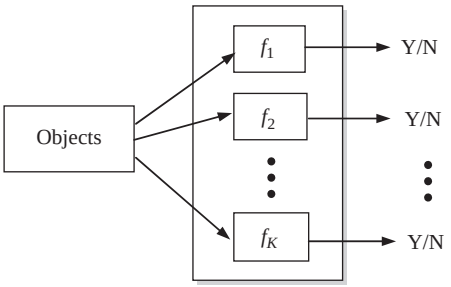

Fig. 1. A function-based classifier.

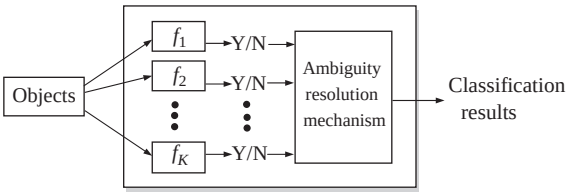

Fig. 2. A function-based classifier with ambiguity resolution.

assigned to the class with the maximum output. The other methods include distance-based classifiers and evolutionary approaches. Distance-based classifiers, like maximum likelihood classifier (MLC) [10] and *k*-nearest neighbor classifiers [10,11], evaluate distances among input vectors of objects, and then classify objects into the individual class with the smallest distance. The evolutionary approaches, generally, include genetic algorithm (GA) [12,13] and genetic programming (GP) [14–17]. Genetic algorithm encodes a set of classification rules to be a sequence of bit strings called gene. The evolution operations such as reproduction, crossover and mutation are used to generate the next generation of classification rules with better fitness. The classifier with a set of classification rules will be obtained after the specified number of generations is evolved or the condition of the fitness function is satisfied. For genetic programming, there are two types of classifiers can be learned from the training data set. The first type is the rule-based classifier consisting of classification rules as other methods [14]. The other type is the function-based classifier in which the discriminant functions are included [15,16]. In the function-based classifier, as Fig. 1, each predefined class has a corresponding discriminant function to decide whether an object belongs to the class or not. The advantages of a function-based classifier are concise and efficient, because each class has only one corresponding function and the functions are easy to compute. Nevertheless, thereare a few problems for learning discriminant functions using genetic programming and classifying using discriminant functions. One of the main drawbacks of using GP methodology is the long training time. Although the more training time allowing the more accurate the classifier can be trained, it is still relatively long while comparing with other classification methods. Another problem is the ambiguity that may occur when a new object is recognized by two or more discriminant functions at the same time or no discriminant function recognizes the object. To resolve the problem of ambiguity, an effective discerning mechanism should be provided, as Fig. 2; otherwise, the accuracy of classification will be decayed. Kishore proposed the strength of association (SA) measure to solve the problem of ambiguity by classifying an ambiguous object to the major class of the corresponding discriminant function; however, the accuracy was not improved so much by SA measure [16].

In this paper, we present a new scheme for learning discriminant functions based on genetic programming and an effective ambiguity resolution mechanism for the discriminant functions of the classifier. In order to shorten the training time of classifiers without losing accuracy of classification, we propose an adaptive incremental learning strategy and the distance-based fitness functions. The proposed learning strategy partitions the training data of each class into several small subsets with both positive and negative training samples in them first. The learning algorithm then learns a specific discriminant function for each class from the sample subsets stage by stage incrementally. After all discriminant functions are generated, we provide the resolution mechanism called Z-value measure to resolve the problem of ambiguity. Two types of distance-based fitness functions: boundary division and interval division and two Z-value ambiguity resolutions: the Algorithm Z and the Algorithm *Z_Min* are proposed. Based on the proposed fitness functions and ambiguity resolutions, four alternative classifiers are produced. Several benchmarks of data sets from UCI data repository are used to demonstrate and compare the accuracy of the proposed methods. We will discuss the effectiveness of the GP learning strategies for distinct fitness functions and the suitability of various Z-value ambiguity resolutions for the learned discriminant functions. The experiments show that a well-designed fitness functions used in GP can improve the training time as well as the accuracy of classification. In comparison with other classification methods, the classifiers learned by the fitness function of interval division including *IZ* and *IZ_min* have high accuracy of classification.

The remainder of this paper is organized as follows. Section 2 reviews the methodology of genetic programming and gives the algorithm in detail by steps. In Section 3, we propose a GP-based adaptive incremental learning approach and the distance-based fitness functions for learning a set of discriminant functions. The ambiguity resolution mechanisms, Z-value measure, and the classification algorithms are presented in Section 4. Section 5 shows the experimental results and makes some comparisons with the previous methods. Finally, we make conclusions and discuss some prospects for future research.

2. Genetic programming

The technique of genetic programming (GP) was proposed by Koza [18,19]. Genetic programming has been applied to a wide range of areas, such as symbolic regression, robot control programs and classifications, etc. Genetic programming can discover underlying data relationships and presents these relationships by expressions. The expression is constructed by terminals and functions. There are several types of functions can be applied for 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.

Genetic programming begins with a set of randomly created individuals called population. Each individual is a potential solution represented as a binary tree. Each binary tree is constructed by all possible compositions of the sets of functions and terminals. A suitable fitness function should be given for evaluating the fitness value of each individual. Then, a set of individuals with better fitness value will be selected and used to generate new population of next generation by the predefined genetic operators. The purpose of genetic operators is used to evolve individuals. The main operators generally include reproduction, crossover, and mutation.

The reproduction operator is the simplest one. This operator copies the individuals with better fitness values to be the population of the next generation directly. Thus, the individuals with better fitness values can be kept continuously in offspring by the reproduction operator. The crossover operator needs more action to generate new individuals. First, two individuals are picked out as parents. Then two sub-trees are randomly selected from the parents, respectively, and swapped each other. After that, two new individuals are generated. For example, there are two individuals $(5+X)+X$ and $(X+X)-2$ in Fig. 3. After the crossover operator is executed, two new individuals, $(X+X)+X$ and $(5+X)-2$, are generated. The mutation operator is usually used for avoiding local optimum. There are two types of mutation operators: Single-node mutation and sub-tree mutation. Single-node mutation is that a terminal or a function in an individual will be replaced by another terminal or function. The other type of mutation, sub-tree mutation, does the same operation with a sub-tree instead of a terminal or a function. For example, in Fig. 4, an individual $(5+X)+X$ becomes $(7+X)$ by sub-tree mutation operator.

After the evolution of a number of generations, a set of individuals with fine fitness value is usually contained in the population. We can take one of the individuals with the best fitness value as the requested result. However, if the fitness values still do not satisfy the specified condition, the process

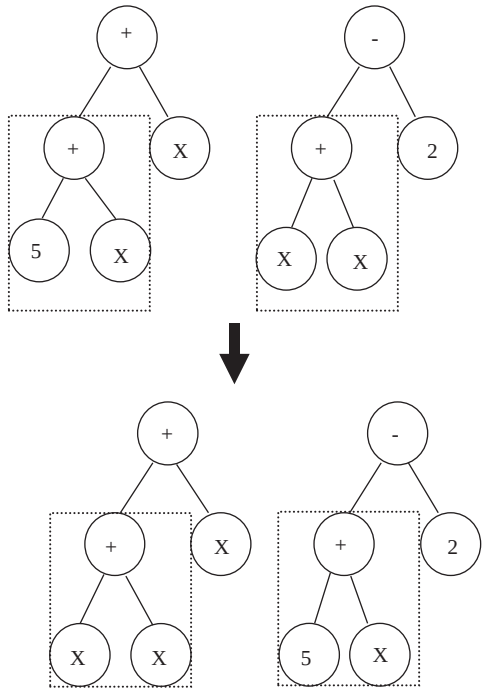

Fig. 3. An example of crossover operation.

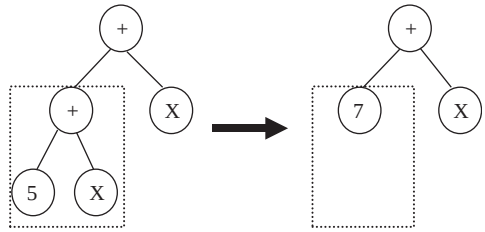

Fig. 4. An example of sub-tree mutation.

of evolution could be continued until the specified condition successes or the number of evolving generations is reached.

We describe the main steps of genetic programming as follows:

Algorithm. Native genetic programming for objective functions learning

Input: The training set T.

Output: The target function with the best fitness value.

Step 1: Initialize the population.

Let $gen = 1$ and we generate the set of initial individuals $\Omega_1 = \{h_1^1, h_2^1, \ldots, h_q^1\}$, where $\Omega_{(gen)}$ is the population in the generation *gen* and $h_i^{(gen)}$ stands for the ith individual in the set $\Omega_{(gen)}$ of the generation *gen*.

Step 2: Evaluate the fitness value of each individual in the training set.

For all $h_i^{(gen)} \in \Omega_{(gen)}$, we compute their fitness values $E_i^{(gen)} = fitness(h_i^{(gen)}, T)$, where the fitness evaluating function *fitness*() is dependent on the problem and is defined by the user.

Step 3: 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 individual $h_i^{(gen)}$ with the best fitness value is returned and the algorithm halts; otherwise, $gen = gen + 1$.

Step 4: Generate the next generation of individuals and go to Step 2.
The new population of next generation $\Omega_{(gen)}$ is generated by the ratio of P_r, P_c and P_m, and then goes to Step 2, where P_r, P_c and P_m represent the probabilities of reproduction, crossover and mutation operations, respectively.

3. Learning discriminant functions by genetic programming

This section presents the GP-based learning method of discriminant functions for classification. First, we will give a formal description for discriminant functions. Then, we provide an adaptive incremental training strategy and propose two distance-based fitness functions to learn the discriminant functions for classifiers. We also give a complete example to explain the proposed learning method by a subset of IRIS data set.

3.1. A formal description of discriminant functions for classification

The notations of used symbols and a formal description of discriminant functions for classification are described in this subsection. Given a data set S, there are *n* attributes for each data $x_j \in S$. Let x_j be represented as

$$x_j = (v_{j1}, v_{j2}, \ldots, v_{jt}, \ldots, v_{jn}), \quad 1 \leqslant t \leqslant n,$$

where $v_{jt} \in \mathbf{R}$ stands for the tth attribute of data x_j in S. Let $C = \{C_1, C_2, \ldots, C_K\}$ be the set of K predefined classes, we say that $\langle x_j, c_j \rangle$ is a sample if the data x_j is assigned to a specified class c_j, $c_j \in C$. We then define a training set (T) on a set of samples such that

$$T = \{\langle x_j, c_j \rangle | x_j = (v_{j1}, v_{j2}, \ldots, v_{jn}), c_j \in C, 1 \leqslant j \leqslant m\},$$

where m is the number of samples in T, denoted as $|T| = m$. Let m_i be the number of samples in T belonging to the class C_i, $1 \leqslant i \leqslant K$,

$$m = (m_1 + m_2 + \cdots + m_i + \cdots + m_K).$$

For classifying unknown data into the K predefined classes, a function-based classifier maintains a set of K discriminant functions $\boldsymbol{F}$,

$$\boldsymbol{F} = \{f_i | f_i : \mathbf{R}^n \to \mathbf{R}, 1 \leqslant i \leqslant K\}.$$

Each discriminant function f_i in the classifier is a function mapping from $\mathbf{R}^n$ to $\mathbf{R}$, $1 \leqslant i \leqslant K$. The task of the discriminant function f_i is to determine whether an unknown data x_j belongs to a specified class C_i or not. Since the image of a discriminant function f_i is a set of real numbers, we partition $\mathbf{R}$ into two separate ranges and specify one of the two ranges as the target range of the class C_i. If the value of $f_i(x_j)$ locates in the target range, the data x_j is an instance of the class C_i; otherwise, the x_j does not belong to the class C_i. We give two types of division methods to separate $\mathbf{R}$ into two ranges: the boundary division and the interval division.

The boundary division simply specifies a constant $a \in \mathbf{R}$ to separate the real number into two half areas. For a sample $\langle x_j, c_j \rangle$, the range based on the boundary division can be defined as

$$\begin{array}{ll} f_i(x_j) \geqslant a & \text{if } c_j = C_i, \\ f_i(x_j) < a & \text{if } c_j \neq C_i, \end{array} \quad \text{where } 1 \leqslant i \leqslant K \text{ and } 1 \leqslant j \leqslant m.$$

The interval division, instead of using a constant number to distinguish the target range from the other, defines an interval to represent the target range for specific class C_i. A discriminant function f_i based on the interval division defines two constant $a, b \in \mathbf{R}$ and $b > 0$. For a sample $\langle x_j, c_j \rangle$, the interval for distinguishing the class C_i is defined as follows:

$$\begin{array}{ll} a - b \leqslant f_i(x_j) \leqslant a + b & \text{if } c_j = C_i, \\ f_i(x_j) < a - b \text{ or } f_i(x_j) > a + b & \text{if } c_j \neq C_i \\ \text{where } 1 \leqslant i \leqslant K \text{ and } 1 \leqslant j \leqslant m. & \end{array}$$

A good discriminant function can transform the data x_j belonging to the class C_i to the values $f_i(x_j)$ mapping into the target range effectively. Thus, an effective function-based classifier is built if we could find the set of effective discriminant functions $\boldsymbol{F}$ including all K predefined classes.

3.2. The proposed learning strategy

We first prepare the training set T before the starting of learning procedures. The samples in training set T include positive instances and negative instances usually. Consider a specific class C_i and a sample $\langle x_j, c_j \rangle \in T$, we say that $\langle x_j, c_j \rangle$ is a positive instance if $c_j = C_i$; otherwise, $\langle x_j, c_j \rangle$ is a negative instance, where $1 \leqslant j \leqslant m$, $1 \leqslant i \leqslant K$. After the training set T was prepared, we start the learning procedure using genetic programming. Conventionally, all of the samples in the training set are fed to the learning procedure at a time. However, when the size of the training set is large, the number of evolving generations in genetic programming will increase rapidly and spend much more time relatively if we want to find an effective solution. Thus, for obtaining effective solutions efficiently by genetic programming, we proceed to learn from the training set using an adaptive incremental learning strategy.

The proposed adaptive incremental learning strategy organizes the training sets to be a sequence of subset of T

having $T_1 \subseteq T_2 \subseteq \cdots \subseteq T_s = T$ and the learning procedure is accomplished stage by stage. The ith learning stage trains the subset of samples T_i and runs the evolution operations up to the specified number of generations to find solutions. While a former learning stage completes the training task using a less number of training samples, it propagates the solutions to the latter stages. A latter learning stage inheriting the solutions from the previous stage then starts its learning stage by increasing the number of training samples. Since the learning stage with a larger training set T_{i+1} is based on the solutions of the previous training set T_i and increases only a few training samples into T_i, the solutions for the training set T_{i+1} can be improved efficiently.

In the above training process, if a large-scale number of samples are increased in each stage, an effective solution may not be generated during a learning stage. On the contrary, if the incremental size of samples is too small, it will waste time on getting an effective intermediate solution. For preventing waste of time and guaranteeing the effectiveness of a solution, we have to handle the increment in each learning stage. We use the following parameters to control the effectiveness of each learning stage:

g: The specified number of generations to be evolved for each learning stage.

ρ: The basic incremental rate, $0 \leqslant \rho \leqslant 1$.

α: The adaptive factor, $\alpha \geqslant 1$.

ω: The condition of satisfying the effectiveness of fitness values.

The detailed algorithm for the proposed adaptive incremental learning strategy is described in the following.

Algorithm. Genetic programming with adaptive incremental learning

Input: The training set T, ρ, ω and g.

Output: The function with the best fitness value.

Step 1: Initialize the parameters.

Let $s = 1$, $\alpha = 1$, $m' = 0$, $m = |T|$, $T_0 = \emptyset$ and $T = T^+ \cup T^-$, where T^+ and T^- represent the set of positive instances and the set of negative instances in the training set T, respectively. The s denotes the current learning stage.

Step 2: Is the halting condition of learning satisfied?

If $m' = m$, then the individual $h_i^{(s,gen)}$ with the best fitness value is returned and the algorithm halts. Otherwise, go to Step 3; a new learning stage is started.

Step 3: Prepare the training set T_s for the learning stage s.

Let T_{inc}^+ and T_{inc}^- be the set of incremental positive instances and incremental negative instances, respectively. $|T_{inc}^+|$ is the number of T_{inc}^+ selected from $T^+ - T_{s-1}$; similarly, $|T_{inc}^-|$ is the number of T_{inc}^- negative instances selected from $T^- - T_{s-1}$. Let $|T_{inc}^+| = \lfloor |T^+| \times \rho \rfloor \times \alpha$ and $|T_{inc}^-| = \lfloor |T^-| \times \rho \rfloor \times \alpha$. Then, we have $T_s = T_{s-1} \cup T_{inc}^+ \cup T_{inc}^-$ and $m' = \min\{m, \lfloor m \times \rho \rfloor \times \alpha + m'\}$.

Step 4: Initialize the population for the learning stage s.

Let $gen = 1$ and generate the set of individuals $\Omega_{s,1} = \{h_1^{s,1}, h_2^{s,1}, \ldots, h_q^{s,1}\}$, where $\Omega_{(s,gen)}$ is the population of the generation *gen* in the stage s and $h_i^{(s,gen)}$ stands for the ith individual of the generation *gen* in stage s.

Step 5: Evaluate the fitness value for the training set T_s in the learning stage s.

For all $h_i^{(s,gen)} \in \Omega_{(s,gen)}$, compute the fitness values $E_i^{(s,gen)} = \mathit{fitness}(h_i^{(s,gen)}, T_s)$, where the fitness evaluating function *fitness*() defined by the user depends on the problem.

Step 6: Is the current learning stage completed?

Case 1: The best fitness value of $E_i^{(s,gen)}$ satisfies the condition ω.

The individual $h_i^{(s,gen)}$ with the best fitness value is propagated to the next stage and double the incremental samples. That is, let $\alpha = 2$, $s = s + 1$ and go to Step 2. While two or more individuals satisfy the condition ω, the shortest one is treated as the best individual.

Case 2: The best fitness value of $E_i^{(s,gen)}$ does not satisfies the condition ω.

If $gen > g$, the individual $h_i^{(s,gen)}$ with the best fitness value is propagated to the next stage but the increment is unchanged; that is, let $\alpha = 1$, $s = s+1$ and go to Step 2. Otherwise, the current stage is to be continued, let $gen = gen + 1$ and go to Step 7.

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

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

3.3. The distance-based fitness functions

In the above algorithm, the fitness evaluating function, $fitness(h_i^{(s,gen)}, T_s)$, is used to measure the effectiveness of an individual $h_i^{(s,gen)}$'s behaving on the training set T_s. A good fitness evaluating function should be able to measure individuals precisely so as to improve not only efficiency but also effectiveness of the learning process. Based on the two range division methods presented in Section 3.1: the boundary division and the interval division, we propose two corresponding distance-based fitness evaluating functions for learning the discriminant functions of a classifier.

For the boundary division, considering a discriminant function f_i for recognizing the class C_i in a classifier, we give a specified constant a and urge that $f_i(x_j) \geqslant a$ if the data x_j belongs to the class C_i; at the same time, we urge $f_i(x_j) \leqslant a$ if the data x_j does not belong to the class C_i, for all $x_j \in T_s$. Hence, for directing an individual $h_i^{(s,gen)}$ in $\Omega_{(s,gen)}$ to achieve the goal of the boundary division, we define the enforcement parameter p and let $p > a$. If a sample $\langle x_j, c_j \rangle$ is a positive instance and $h_i^{(s,gen)}(x_j) \geqslant a$, is a

correct result and no error. However, when $h_i^{(s,gen)}(x_j) < a$ for a positive instance $\langle x_j, c_j\rangle$, it results a wrong classification. At this time, the error of the sample $\langle x_j, c_j\rangle$ on the individual $h_i^{(s,gen)}$ is measured by the distance of $|h_i^{(s,gen)}(x_j) - a|$ plus $|p - a|$. The distance of $|p - a|$ is used to enlarge the error value while the value $h_i^{(s,gen)}(x_j)$ is close to a. The main purpose is to avoid selecting the individuals with many small distances of $|h_i^{(s,gen)}(x_j) - a|$ as new individuals in the next generation of evolution, since our goal is to learn the function that can separate the ranges between positive instances as far as possible. Thus, the fitness measure for a positive instance is defined as

$$D_p(x_j, c_j) = \begin{cases} 0 & \text{if } c_j = C_i \text{ and } h_i^{(s,gen)}(x_j) \geqslant a, \\ |p - h_i^{(s,gen)}(x_j)| & \text{if } c_j = C_i \text{ and } h_i^{(s,gen)}(x_j) < a. \end{cases} \quad (1)$$

Similarly, the fitness measure for a negative instance is defined as

$$D_n(x_j, c_j) = \begin{cases} |h_i^{(s,gen)}(x_j) + p| & \text{if } c_j \neq C_i \text{ and } h_i^{(s,gen)}(x_j) \geqslant a, \\ 0 & \text{if } c_j \neq C_i \text{ and } h_i^{(s,gen)}(x_j) < a. \end{cases} \quad (2)$$

The fitness evaluating function of an individual $h_i^{(s,gen)}$ for the training set T_s in stage s is defined to be

$$Fitness(h_i^{(s,gen)}, T_s) = -\sum_{j=1}^{m'} (D_p(x_j, c_j) + D_n(x_j, c_j)), \quad (3)$$

where m' is the number of training samples in the current learning stage s, $\langle x_j, c_j\rangle \in T_s$, $1 \leqslant j \leqslant m'$.

For the interval division, considering a discriminant function f_i for recognizing the class C_i in a classifier, we give two constants a and b and urge that $|f_i(x_j) - a| \leqslant b$ if the data x_j is a positive instance; at the same time, we urge $|f_i(x_j) - a| > b$ if the data x_j is a negative instance, for all $x_j \in T_s$. That is, the fitness values of positive instances should be located in the specified interval $[a - b, a + b]$. As the same reason of the boundary division, we also give the enforcement parameter p, $p > 0$, for pushing an individual $h_i^{(s,gen)}$ in $\Omega_{(s,gen)}$ to achieve the goal of the interval division. If a sample $\langle x_j, c_j\rangle$ is a positive instance and $|h_i^{(s,gen)}(x_j) - a| \leqslant b$, it is a correct result and no error. However, when $|h_i^{(s,gen)}(x_j) - a| > b$ for a positive instance $\langle x_j, c_j\rangle$, it results a wrong classification. The fitness measure for a positive instance and a negative instance is respectively defined as follows:

$$D_p(x_j, c_j) = \begin{cases} 0 & \text{if } c_j = C_i \text{ and } |h_i^{(s,gen)}(x_j) - a| \leqslant b, \\ |p + (|h_i^{(s,gen)}(x_j) - a| - b)| & \text{if } c_j = C_i \text{ and } |h_i^{(s,gen)}(x_j) - a| > b, \end{cases} \quad (4)$$

$$D_n(x_j, c_j) = \begin{cases} |p - (|h_i^{(s,gen)}(x_j) - a| - b)| & \text{if } c_j \neq C_i \text{ and } |h_i^{(s,gen)}(x_j) - a| \leqslant b, \\ 0 & \text{if } c_j \neq C_i \text{ and } |h_i^{(s,gen)}(x_j) - a| > b. \end{cases} \quad (5)$$

The fitness value of an individual $h_i^{(s,gen)}$ for the training set T_s in stage s is defined to be

$$Fitness(h_i^{(s,gen)}, T_s) = -\sum_{j=1}^{m'} (D_p(x_j, c_j) + D_n(x_j, c_j)), \quad (6)$$

where m' is the number of training samples in the current learning stage s, $\langle x_j, c_j\rangle \in T_s$, $1 \leqslant j \leqslant m'$.

Since we use the negative of measure to be the fitness value of an individual, the best fitness value is zero. While the fitness value of the individual $h_i^{(s,gen)}$ is zero, it means that the function $h_i^{(s,gen)}$ can discriminate all samples of the class C_i from the others in the given m' training samples. Hence, we can choose the individual $h_i^{(s,gen)}$ as the discriminant function f_i for the class C_i.

3.4. An example

For illustrating the proposed adaptive incremental learning approaches, we give an example with the boundary division to show the learning of discriminant functions in the following.

Example 1. In the Fisher's Iris dataset [20], there are four numerical attributes in each object: sepal length, sepal width, petal length, and petal width. We denote these attributes as *SL*, *SW*, *PL*, and *PW*, respectively. In this example, we take a subset of 12 objects from the original Iris dataset as the training set T, as Table 1 That is, $|T| = m = 12$. We now begin the adaptive incremental learning procedure to find the discriminant function f_{Setosa} for the *Setosa* class as follows:

Input: The training set T, $\rho = 0.3$, ω is "the fitness value equals to 0", and $g = 100$.

Step 1: Initialize the parameters. We initially set $s = 1$, $\alpha = 1$, $m' = 0$, $m = 12$, and $T_0 = \emptyset$ for the learning

Table 1
The 12 training data T for Example 1

Data	*SL*	*SW*	*PL*	*PW*	Class
D_1	6.3	2.7	4.9	1.8	*Virginica*
D_2	4.9	3.0	1.4	0.2	*Setosa*
D_3	6.7	3.1	4.4	1.4	*Versicolor*
D_4	5.1	3.5	1.4	0.2	*Setosa*
D_5	6.4	3.1	5.5	1.8	*Virginica*
D_6	5.6	3.0	4.1	1.3	*Versicolor*
D_7	5.0	3.6	1.4	0.2	*Setosa*
D_8	7.2	3.2	6.0	1.8	*Virginica*
D_9	6.7	3.1	4.7	1.5	*Versicolor*
D_{10}	6.1	3.0	4.9	1.8	*Virginica*
D_{11}	6.0	3.4	4.5	1.6	*Versicolor*
D_{12}	5.2	3.4	1.4	0.2	*Setosa*

algorithm. Moreover, we set $a = 0$ and $p = 10$ for the fitness function of the boundary division. The set of positive instances of the training set T in Table 1 for the *Setosa* class is

$$T^+ = \{D_2, D_4, D_7, D_{12}\},$$

and the set of negative instances is

$$T^- = \{D_1, D_3, D_5, D_6, D_8, D_9, D_{10}, D_{11}\}.$$

Step 2: Since $m' \neq m$, a new stage, Stage 1, is started and goes to Step 3.

Stage 1

Step 3: Prepare the training set T_1 for the first learning stage.

$$|T^+_{inc}| = \lfloor |T^+| \times \rho \rfloor \times \alpha = \lfloor 4 \times 0.3 \rfloor \times 1 = 1,$$
$$|T^-_{inc}| = \lfloor |T^-| \times \rho \rfloor \times \alpha = \lfloor 8 \times 0.3 \rfloor \times 1 = 2.$$

Assume that

$$\begin{aligned} T_1 &= T_0 \cup T^+_{inc} \cup T^-_{inc} \\ &= \emptyset \cup \{D_2\} \cup \{D_1, D_3\} \\ &= \{D_1, D_2, D_3\}. \end{aligned}$$

Let T_1 be selected and $m' = \min\{m, \lfloor m \times \rho \rfloor \times \alpha + m'\} = \min\{12, \lfloor 12 \times 0.3 \rfloor \times 1 + 0\} = 3$.

Step 4: Let $gen = 1$ and generate the set of individuals $\Omega_{1,1} = \{h^{1,1}_1, h^{1,1}_2, \ldots, h^{2,1}_q\}$.

Step 5: Evaluate the fitness values $E^{1,gen}_i$ on the training set T_1 in the Stage 1.

Steps 5–7 are repeated until the best fitness value of $E^{1,gen}_i$ satisfies the condition of $fitness(h^{1,gen}_i, T_1) = 0$ or $gen > 100$.

Step 6: Assume that $gen > 100$ at this stage, the individual with best fitness value after 100 generations is

$$h^{1,100}_i = SL - PL - PW,$$

Table 2
The fitness values of the individual $h^{1,100}_i = SL - PL - PW$

Data	*h* Value	$D_n \vert D_p$	Class
D_1	−0.4	$D_n = 0$	*Virginica*
D_2	3.3	$D_p = 0$	*Setosa*
D_3	0.9	$D_n = 10.9$	*Versicolor*

and the fitness value of the individual is

$$-\sum_{j=1}^{3} (D_p + D_n) = -10.9.$$

Since the fitness value is not equal to zero after 100 generations, we set $\alpha = 1$ and $s = 2$, and the procedure goes to Step 2. The values of D_n and D_p of the training data in T_1 are listed in Table 2 .

Step 2: Since $m' = 3 \neq m = 12$, go to Step 3, Stage 2 is started.

Stage 2

Step 3: Prepare the training set T_2 for the Stage 2. We have

$$|T^+_{inc}| = \lfloor |T^+| \times \rho \rfloor \times \alpha = \lfloor 4 \times 0.3 \rfloor \times 1 = 1,$$
$$|T^-_{inc}| = \lfloor |T^-| \times \rho \rfloor \times \alpha = \lfloor 8 \times 0.3 \rfloor \times 1 = 2.$$

Assume that $T^+_{inc} = \{D_4\}$ and $T^-_{inc} = \{D_5, D_6\}$ are selected from $T - T_1$, thus

$$\begin{aligned} T_2 &= T_1 \cup T^+_{inc} \cup T^-_{inc} \\ &= \{D_1, D_2, D_3\} \cup \{D_4, D_5, D_6\} \\ &= \{D_1, D_2, D_3, D_4, D_5, D_6\}, \end{aligned}$$

and $m' = \min\{12, \lfloor 12 \times 0.3 \rfloor \times 1 + 3\} = 6$.

Step 4: Let $gen = 1$ and generate the set of individuals $\Omega_{2,1} = \{h^{2,1}_1, h^{2,1}_2, \ldots, h^{2,1}_q\}$.

Step 5: Evaluate the fitness on the training set T_2 in Stage 2.

Steps 5–7 are repeated until the best fitness value of $E^{2,gen}_i$ satisfies the condition of $fitness(h^{2,gen}_i, T_2) = 0$ or $gen > 100$.

Step 6: Assume that there is an individual satisfies the condition of $fitness(h^{2,50}_i, T_2) = 0$ while $gen = 50$:

$$h^{2,50}_i = SL - PL - PW - PW.$$

The fitness value of the individual is

$$-\sum_{j=1}^{6} (D_p + D_n) = 0.$$

Since the condition ω is satisfied, we set $\alpha = 2$, $s = 3$ and the procedure goes to Step 2. The values of D_n and D_p of the training data are shown in Table 3 .

Step 2: Since $m' = 6 \neq m = 12$, go to Step 3, and Stage 3 is started.

Table 3
The fitness values of the individual $h_i^{2,50} = SL - PL - PW - PW$

Data	Value h	$D_n \mid D_p$	Class
D_1	−2.2	$D_n = 0$	*Virginica*
D_2	3.1	$D_p = 0$	*Setosa*
D_3	−0.5	$D_n = 0$	*Versicolor*
D_4	3.3	$D_p = 0$	*Setosa*
D_5	−2.7	$D_n = 0$	*Virginica*
D_6	−1.1	$D_n = 0$	*Versicolor*

Stage 3

Step 3: Prepare the training set T_3 for the Stage 3. We have

$$|T_{inc}^{+}| = \lfloor |T^{+}| \times \rho \rfloor \times \alpha = \lfloor 4 \times 0.3 \rfloor \times 2 = 2,$$
$$|T_{inc}^{-}| = \lfloor |T^{-}| \times \rho \rfloor \times \alpha = \lfloor 8 \times 0.3 \rfloor \times 2 = 4,$$

Assume that $T_{inc}^{+} = \{D_7, D_{12}\}$ and $T_{inc}^{-} = \{D_8, D_9, D_{10}, D_{11}\}$ are selected from $T - T_2$, thus

$$\begin{aligned} T_3 &= T_2 \cup T_{inc}^{+} \cup T_{inc}^{-} \\ &= \{D_1, D_2, D_3, D_4, D_5, D_6\} \cup \{D_7, D_{12}\} \\ &\quad \cup \{D_8, D_9, D_{10}, D_{11}\} \\ &= \{D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, \\ &\quad D_{10}, D_{11}, D_{12}\}, \end{aligned}$$

and $m' = \min\{m, \lfloor m \times \rho \rfloor \times \alpha + m'\} = \min\{12, \lfloor 12 \times 0.3 \rfloor \times 2 + 6\} = 12$.

Step 4: Let $gen = 1$ and generate the set of individuals $\Omega_{3,1} = \{h_1^{3,1}, h_2^{3,1}, \ldots, h_q^{3,1}\}$.

Step 5: Evaluate the fitness on the training set T_3 in Stage 3.

Steps 5–7 are repeated until the best fitness value of $E_i^{3,gen}$ satisfies the condition of $fitness(h_i^{3,gen}, T_3) = 0$ or $gen > 100$.

Step 6: Assume that there is an individual satisfies the condition of $fitness(h_i^{3,20}, T_3) = 0$ while $gen = 20$:

$$h_i^{3,20} = SW - PL,$$

and the fitness value of the individual is

$$-\sum_{j=1}^{12} (D_p + D_n) = 0.$$

Since the condition ω is satisfied, we set $\alpha = 2$, $s = 4$ and the procedure goes to Step 2. The values of D_n and D_p of the training data are shown in Table 4 .

Step 2: Since $m' = m = 12$, the individual $SW - PL$ is returned and the algorithm halts. The individual will be used to be the discriminant function of the *Setosa* class f_{Setosa}.

By the same learning procedure, we can obtain the classifier $\boldsymbol{F} = \{f_{Setosa}, f_{Versicolor}, f_{Virginica}\}$. In this case,

Table 4
The fitness values of the function $h_i^{(3,20)} = SW - PL$

Data	h Value	$D_n \mid D_p$	Class
D_1	−2.2	$D_n = 0$	*Virginica*
D_2	1.6	$D_p = 0$	*Setosa*
D_3	−1.3	$D_n = 0$	*Versicolor*
D_4	2.1	$D_p = 0$	*Setosa*
D_5	−2.4	$D_n = 0$	*Virginica*
D_6	−1.1	$D_n = 0$	*Versicolor*
D_7	2.2	$D_p = 0$	*Setosa*
D_8	−2.8	$D_n = 0$	*Virginica*
D_9	−1.6	$D_n = 0$	*Versicolor*
D_{10}	−1.9	$D_n = 0$	*Virginica*
D_{11}	−1.1	$D_n = 0$	*Versicolor*
D_{12}	2.0	$D_p = 0$	*Setosa*

Table 5
The values of training data for the discriminant functions

Data	f_{Setosa}	$f_{Versicolor}$	$f_{Virginica}$	Class
D_1	−2.2	−68.0944	20.9091	*Virginica*
D_2	1.6	−0.8493	−2.7187	*Setosa*
D_3	−1.3	6.1413	−20.1754	*Versicolor*
D_4	2.1	−0.8327	−2.7316	*Setosa*
D_5	−2.4	−6.7874	20.5357	*Virginica*
D_6	−1.1	5.9744	−11.8557	*Versicolor*
D_7	2.2	−0.8054	−2.7251	*Setosa*
D_8	−2.8	−15.2502	17.9688	*Virginica*
D_9	−1.6	8.8143	−41.0714	*Versicolor*
D_{10}	−1.9	−22.9747	21.6981	*Virginica*
D_{11}	−1.1	11.5634	−191.6669	*Versicolor*
D_{12}	2.0	−0.8607	−2.7381	*Setosa*

the following three discriminant functions are learned:

$$f_{Setosa} = SW - PL,$$

$$f_{Versicolor} = \frac{PL \times SL - \frac{PL}{-119} - \frac{-4 \times (PW - SL)}{PL}}{\frac{PW \times SL}{PL - SL} + 5 + SW + \frac{-16}{-104}},$$

$$f_{Virginica} = \frac{115}{29 \times PW + SL - 53}.$$

The values of training data with the three discriminant functions are shown in Table 5

4. The Z-value measure and the classification methods

In general, since the training set does not consist of all possible samples, a classifier cannot recognize all objects correctly in real applications. The traditional rule-based classifiers need high accurate rules to achieve the effectiveness of recognition. However, the recognition rate of the proposed function-based classifier is dependent on not only

one discriminant function itself but also others discriminant functions in the classifier, since the misjudgment of a discriminant function will make the classifier misclassified or ambiguous. Misclassification occurs only when an object is recognized by a single discriminant function in the classifier and the recognized class is a wrong one. The ambiguous cases are more complicated than misclassification, thus we discuss the problem of ambiguity in the following two situations:

Case 1: *Conflict*. An unknown data is recognized by two or more discriminant functions in the classifier at the same time.

Case 2: *Rejection*. An unknown data is recognized by no discriminant function in the classifier. Generally, the probability of misclassification is much less than ambiguity. Hence, for improving the recognition rate of a function-based classifier, an effective ambiguity resolution mechanism is needed. Here, we propose Z-value measure to handle the problem of ambiguity. The Z-value is defined and described in the following.

Let T_{C_i} be the set of positive instances for class C_i belonging to the training set T, $T_{C_i} = \{x_j | \langle x_j, c_j \rangle \in T$ and $c_j = C_i, 1 \leqslant j \leqslant m\}$ and $|T_{C_i}| = m_i$. We consider a class C_i and its corresponding discriminant function $f_i \in \boldsymbol{F}$, $1 \leqslant i \leqslant K$. We define that μ_i is the mean of values of $f_i(x_j)$ for $x_j \in T_{C_i}$. That is,

$$\mu_i = \frac{\sum_{x_j \in T_{C_i}} f_i(x_j)}{m_i}, \quad 1 \leqslant i \leqslant K. \tag{7}$$

For each μ_i, the standard deviation of values of $f_i(x_k)$, $x_k \in T_{C_i}$, is defined as

$$\sigma_i = \sqrt{\frac{\sum_{x_j \in T_{C_i}} (f_i(x_j) - \mu_i)^2}{m_i}}, \quad 1 \leqslant i \leqslant K. \tag{8}$$

Now, for an n-attribute dataset S, let $x_k \in S$ and a discriminant function $f_i \in \boldsymbol{F}$, $1 \leqslant i \leqslant K$, the Z-value of data x_k for f_i is defined as

$$Z_i(x_k) = \frac{|f_i(x_k) - \mu_i|}{\sigma_i}, \tag{9}$$

where $x_k \in S$, $1 \leqslant i \leqslant K$. If the data x_k is only recognized by a unique discriminant functions f_i in $\boldsymbol{F}$, the x_k is assigned to the class C_i and the task of classification is done. However, once a data is not recognized by any discriminant function or a data is recognized by two or more discriminant functions in $\boldsymbol{F}$, the mechanism of Z-value measure is applied to resolve the problem of ambiguity. For a discriminant function f_i and a data x_k, since the Z-value of x_k for f_i, $Z_i(x_k)$, represents the variance between the data x_k and the class C_i, this variance can be used to decide the class to which the x_k should belongs. We propose two classification methods based on the Z-value measure, the Algorithm Z and the Algorithm Z_min. The detailed algorithms are shown in the following.

Algorithm Z. **Classification by the** Z**-value measure on ambiguous cases**

Input: An unknown data x_k and the set $\boldsymbol{F}$ of K learned discriminant functions.

Output: the assigned class C_l for x_k

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

Step 2: If the data x_k is recognized by f_i, then $\boldsymbol{Z} = \{f_i\} \cup \boldsymbol{Z}$.

Step 3: If $i < K$, then $i = i + 1$, go to Step 2; otherwise, go to Step 4.

Step 4: Let $|\boldsymbol{Z}|$ be the number of functions in $\boldsymbol{Z}$. If $|\boldsymbol{Z}| = 1$, the unique class C_l corresponding to the function f_l in $\boldsymbol{Z}$ will be returned and stop; otherwise, go to Step 5.

Step 5: If $|\boldsymbol{Z}| = 0$, $\boldsymbol{Z} = \boldsymbol{F}$.

Step 6: Compute all $Z_i(x_k)$, where $f_i \in \boldsymbol{Z}$.

Step 7: Find the $l = \arg\min_{f_i \in Z}\{Z_i(x_k)\}$ and assign the data x_k to the class C_l.

Algorithm Z_min. **Classification by the minimum** Z**-value measure**

Input: An unknown data x_k and the set $\boldsymbol{F}$ of K learned discriminant functions.

Output: the assigned class C_l for x_k

Step 1: Compute all $Z_i(x_k)$, where $f_i \in \boldsymbol{F}$.

Step 2: Find the $l = \arg\min_{f_i \in F}\{Z_i(x_k)\}$ and assign the data x_j to the class C_k.

For cases of conflict, the mechanism used in the Algorithm Z collects the conflicting discriminant functions on data x_k into the set of $\boldsymbol{Z}$. Then, we compute the Z-value $Z_i(x_k)$ on each discriminant function in $\boldsymbol{Z}$ and assign data x_k to the class with the smallest Z-value. For the case of rejection, since there is no discriminant function in the set of $\boldsymbol{Z}$, the Algorithm Z puts all discriminant functions in $\boldsymbol{F}$ into $\boldsymbol{Z}$ and resolves the case as all discriminant functions conflict. However, the algorithm Z_min does not recognize data x_k by individual discriminant function in the classifier $\boldsymbol{F}$. Instead of resolving ambiguity as in the Algorithm Z, the algorithm Z_min directly assigns the data x_k to the class with the smallest Z-value. If the number of the smallest Z-values is more than one, although the probability of such situation is very small, we dedicate the data to the major class when once it happens.

Example 2. We use the discriminant functions f_{Setosa}, $f_{Versicolor}$, and $f_{Virginica}$ obtained from Example 1 to construct a classifier and classify the test data. Let $\boldsymbol{F} = \{f_{Setosa}, f_{Versicolor}, f_{Virginica}\}$, the μ_i and σ_i of the discriminant functions f_{Setosa}, $f_{Versicolor}$, and $f_{Virginica}$ can be calculated from Table 5, as follows:

$$\mu_{Setosa} = \frac{(1.6 + 2.1 + 2.2 + 2)}{4} = 1.975,$$

Table 6
The test data set for Example 2

Data	*SL*	*SW*	*PL*	*PW*	Real class	f_{Setosa}	$f_{Versicolor}$	$f_{Virginica}$	Assigned class
TD_1	5.1	3.8	1.6	0.2	*Setosa*	2.2	−0.4791	−2.7316	*Setosa*
TD_2	4.6	3.2	1.4	0.2	*Setosa*	1.8	−0.7734	−2.6995	*Setosa*
TD_3	5.3	3.7	1.5	0.2	*Setosa*	2.2	−0.6694	−2.7446	*Setosa*
TD_4	5.0	3.3	1.4	0.2	*Setosa*	1.9	−0.8355	−2.7251	*Setosa*
TD_5	6.7	3.0	5.0	1.7	*Versicolor*	−2.0	22.7246	38.3333	*Versicolor* and *Virginica*
TD_6	6.2	2.9	4.3	1.3	*Versicolor*	−1.4	6.0521	−12.6374	*Versicolor*
TD_7	5.1	2.5	3.0	1.1	*Versicolor*	−0.5	2.0693	−7.1875	*Versicolor*
TD_8	5.4	3.0	4.5	1.5	*Versicolor*	−1.5	−20.8712	−28.0488	None
TD_9	6.8	3.2	5.9	2.3	*Virginica*	−2.7	−4.0444	5.6098	*Virginica*
TD_{10}	6.5	3.0	5.2	2.0	*Virginica*	−2.2	−15.1911	10.0000	*Virginica*
TD_{11}	6.2	3.4	5.4	2.3	*Virginica*	−2.0	−3.2506	5.7789	*Virginica*
TD_{12}	5.9	3.0	5.1	1.8	*Virginica*	−2.1	−5.1028	22.5490	*Virginica*

$$\mu_{Versicolor} = \frac{(6.1413 + 5.9744 + 8.8143 + 11.5634)}{4} = 8.1234,$$

$$\mu_{Virginica} = \frac{(20.9091 + 20.5357 + 17.9688 + 21.6981)}{4} = 20.2779.$$

$$\sigma_{Setosa} = 0.2278,$$

$$\sigma_{Versicolor} = 2.2835,$$

$$\sigma_{Virginica} = 1.3977.$$

After μ_i and σ_i are generated, assume that we have 12 test data denoted as $\{TD_1, TD_2, \ldots, TD_{12}\}$, which is shown in the columns of *SL*, *SW*, *PL*, and *PW* in Table 6 . The values of the corresponding discriminant functions are listed as the columns of f_{Setosa}, $f_{Versicolor}$, and $f_{Virginica}$ in Table 6, respectively. Owing to the discriminant functions in Example 1 are learned using the boundary division, while we apply the Algorithm Z to classify the test data in Table 6, the test data will be assigned to the class whose function value is larger than or equal to zero. In Table 6, we found that most of the test data can be assigned to the correct class except the TD_5 and the TD_8. The TD_5 is recognized by $f_{Versicolor}$ and $f_{Virginica}$ at the same time; thus, the conflict occurs in this case. The Algorithm Z resolves this conflict case by calculating the values of $Z_{Versicolor}$ and $Z_{Virginica}$ and assigns the TD_5 to the class of *Versicolor*, since $Z_{Versicolor}$ is less than $Z_{Virginica}$ as shown in the first row of Table 7 . For the TD_8, it is a case of rejection because no discriminant function recognizes such object. The Algorithm Z must calculate the Z-values of all discriminant functions for the TD_8. The second row in Table 7 shows that $Z_{Versicolor} = 12.6973$ is the smallest, hence, the TD_8 is assigned to the class of *Versicolor*, too.

While the Algorithm *Z_min* is applied to classify the data, we will calculate the Z-values of all discriminant functions for each TD_i and assign the TD_i to the class having the smallest Z-value like the rejection case of the TD_8.

Table 7
The Z-values of TD_5 and TD_8

Data	Z-value of f_{Setosa}	Z-value of $f_{Versicolor}$	Z-value of $f_{Virginica}$
TD_5		6.3942	12.9179
TD_8	15.2572	12.6973	34.5769

5. Experimental results and comparisons

In this section, we demonstrate and compare the performance of the proposed classifiers. The classifiers proposed in this paper consist of sets of discriminant functions learned by genetic programming and the ambiguity resolutions. We refer to the learning methods using the boundary division and the interval division as GP-B and GP-I, respectively. For demonstrating the effectiveness and efficiency of the proposed classifiers, we modify the GP Quick 2.1 [21] to fit the requirements of the proposed approaches and perform the experiments since the source code of GP Quick is well known and easily accessible from the web. The experiments are done by using a PC with 866 MHz CPU and 128 MB RAM.

We select 11 datasets with all numeric attributes as our test datasets from UCI data depository [20], which are well-known benchmark for evaluating the accuracy of classifiers. These selected datasets have many distinct features including the number of attributes, the number of classes and the size of each dataset. All selected datasets are summarized in Table 8 . Some of the datasets containing miss values are modified, such as the Wisconsin breast cancer dataset (bcw) and PIMA Indian diabetes dataset (*pima1* and *pima2*). The original Wisconsin breast cancer dataset contains 699 cases separated into two classes called *Malignant* and *Benign* with

Table 8
The test datasets

Datasets	Number of attributes	Number of cases	Number of classes
bcw	9	683	2
bupa	6	345	2
glass	9	214	7
iris	4	150	3
ionosphere	34	351	2
pima1	8	768	2
pima2	7	532	2
sonar	60	229	2
vehicle	18	846	4
waveform	21	5000	3
wine	13	178	3

241 cases and 458 cases, respectively. Each object in the bcw dataset has nine numerical attributes. However, 16 cases in the bcw dataset consist of missing values. The 683 cases without missing values are used to evaluate classifiers after removing the 16 incomplete data. The remaining dataset contains 239 data of *Malignant* and 444 data of '*Benign*'. The PIMA Indian diabetes dataset contains 768 cases that are separated into two classes: tested positive for diabetes or not. Each case has eight numerical attributes. However, the attribute, namely 2-h serum insulin, contains many zero values, which are physically impossible [22]. Hence, we prepare two versions of the PIMA Indian diabetes dataset: the *pima1* and the *pima2*. The original dataset is denoted as the *pima1*. The *pima1* contains 500 cases in the positive class and 268 cases in the negative class. Then, we remove the attribute of 2-h serum insulin and some records that have impossible values in other attributes from the original dataset. The remaining dataset is denoted as the *pima2*, which contains 532 cases and each case has only seven attributes. In *pima2*, the number of cases in the positive class is 355 and the number of cases in the negative class is 177. The others datasets are unchanged. All datasets is tested in 10-fold cross validation for ten runs [23].

The proposed classifiers first learn the discriminant function f_i for each corresponding class C_i using genetic programming. The parameters used in GP Quick and the adaptive incremental learning are set as Table 9 . The maximum generation for each stage g is set to be 1000, $\rho = 0.2$, and ω is set as "the fitness value equals to 0". The parameters of fitness function for the GP-B are $p = 10$ and $a = 0$, and for the GP-I are $p = 10$, $a = 0$, and $b = 10$. We experience these values by experiments in this paper. The training times of each discriminant function for all selected datasets are shown in Table 10 . The table shows the minimum, the maximum and the average learning time for each discriminant function during the ten runs of 10-fold validation test. After the discriminant functions of a dataset are learned, we test the accuracy for each discriminant function. The accuracy of a discriminant function, *Acc*, is defined to be

$$Acc = \sum_{i=1}^{K} n_i \Big/ \sum_{i=1}^{K} m_i \ ,$$

where m_i is the number of cases in the class C_i, and n_i is the number of cases recognized only by the discrminant function f_i and belonging to the class C_i. The results are shown as the column of *Acc* in Table 11 . Since the ambiguous cases happened, we list the rates of conflict and rejection in the columns of $r_{conflict}$ and r_{reject} of Table 11, respectively. Then, the classification algorithms, the Algorithm Z and the Algorithm *Z_min*, are applied to resolve the ambiguous cases for the discriminant functions learned from the GP-B and the GP-I. The final classification results are shown in the columns of *BZ*, *BZ-min*, *IZ* and *IZ_min* of Table 11. The *BZ* method means using the GP-B to learn the discriminant functions and classifying data by the Algorithm Z. The *BZ_min* method uses the GP-B and the Algorithm *Z_min*. Similarly, the methods of *IZ* and *IZ_min* are the GP-I in combination with the Algorithm Z and the Algorithm *Z_min*, respectively.

The training time of a discriminant function is mainly dependent on datasets and fitness functions used in genetic programming. We discuss the reasons in the following. For the reason of datasets, first, it is obvious that we need spend more training time on learning discriminant functions from a larger dataset. Nevertheless, the training time is independent of the number of attributes in datasets. Secondly, since the adaptive incremental learning will start a new learning stage once the current learning stage gets a good discriminant function, it will take less training time if the instances belonging to the corresponding class are easy to be distinguished from the others in the dataset. For example, we knew that classifying the class *Versicolor* in the iris dataset is more difficult than classifying the class *Setosa*. The training time for $f_{Versicolor}$ is longer than f_{Setosa}. Another example is that although the number of instances in the ionosphere dataset (351 cases) is more than the bupa's (345 cases), the training time of the ionosphere dataset is shorter than the bupa's because of its higher recognition rate. Hence, the difficulty of classification about the classes in datasets will reflect the training time. For another reason of fitness functions, a good fitness function is able to find out effective discriminant functions so that the adaptive incremental learning can be halt earlier. At the same time, the advantage of the adaptive incremental learning strategy can make an effective fitness function be performed more efficiently. Hence, a well-defined fitness function will improve the effectiveness and efficiency of learning process. From Tables 10 and 11, we found that the mean training time of GP-I is shorter than the GP-B's generally, and the accuracy of discriminant functions learned by GP-I is better than the GP-B's for the same class. That is to say, the interval division is more effective and efficient than the boundary division in the learning process.

Table 9
The used parameters of experiments

Parameters	Values	Parameter	Values
Node mutate weight	43.5%	Mutation weight annealing	40%
Mutate constant weight	43.5%	Generations per stage g	1000
Mutate shrink weight	13%	Incremental rate ρ	0.2
Selection method	Tournament	Criterion number ω	Fitness = 0
Tournament size	7	Function set	$+, -, \times, \div$
Crossover weight	28%	Population size	1000
Crossover weight annealing	20%	a, p for GP-B	0, 10
Mutation weight	8%	a, b, p for GP-I	0, 10, 10
Max tree depth	7		

To explain the reason why the interval division is better, we should discuss the properties of their fitness functions used first. The difference between the boundary division and the interval division is the mapped ranges of positive instances. The boundary division maps the positive instances to an unlimited half area. While the adaptive incremental learning strategy is learning, the function learned from the initial subset in the training set may be too loose to map the most part of the training set to the correct area. For instance, the learned function only contains one of the attributes that can distinguish positive instances and negative instances by multiplying a minus. However, this rule may be true only for the subset of training data in initial learning stages. While the training data is increasing, such function is not fit to the next learning stage any more. Thus, the larger training set must take more time for evolving accurate discriminant functions. On the contrary, the interval division maps the positive instances to a limited interval. Any positive instance located outside the interval or negative instance located inside the interval will produce error. Thus, the learned function must consider more attributes in order to represent the characteristics of the initial subset of the training set. Such function, generally, will be conducive to evolve an effective discriminant function for the complete training set in later learning stages efficiently. As Table 11 shows, the accuracy of the discriminant functions learned by GP-I are better than GP-B's for most of the datasets except the bcw dataset and the bupa dataset. However, the decreases of accuracy are small (*Acc*: −1.5% for the bcw and −0.9% for the bupa) and the main reason of decreasing is not caused by the decreasing rate of rejection (r_{reject}: −0.3% for the bcw and −0.5% for the bupa) but the increasing rate of conflict ($r_{conflict}$: +1.7% for the bcw and +2.9% for the bupa). It means that the discriminant functions learned by GP-I still can recognize more cases than the GP-B's though some cases may conflict.

From the above explanation, also, we can easily realize that the results of the classification method *BZ* will be better than the method *BZ_min*. Since GP-B only urges the training set to map to two half unlimited area, the function values of some exceptions or noise in the training set may be too large or too small. It results in interference of the computation on μ_i and σ_i. The confidence of Z-values thus is relatively low in comparison with the boundary condition of the discriminant function. That is to say, the Algorithm Z is more suitable than the Algorithm *Z_min* for GP-B. The experiments in Table 11 show such result that the method *BZ* is more accurate and more stable than the method *BZ_min*. On the other hand, the discriminant functions learned by GP-I restrict the range of positive instances in the training set to be located in a specified interval. In statistical, the mean values of positive instances mapped by a discriminant function will be a normal distribution if the data in training sets reflect uniform sampling of the real domain. Thus, the Z-value $Z_i(x_k)$ of an instance x_k for a discriminant function f_i for the class C_i can be used to represent the degree of the instance x_k belonging to the class C_i. In Table 11, the experiments demonstrate the Z-value can resolve the cases of conflict and rejection effectively in the method *IZ*. Furthermore, we found that the classification method *IZ_min* determining the class of an unknown instance using the Z-values directly is even more accurate than the method *IZ* for most of datasets. The method *IZ_min* is superior to *IZ* in 7 datasets for the selected 11 datasets. Three of them including the bupa (+9.36%), the *pima*1(+8.90%) and the *pima*2(+8.93%) improve a lot especially. However, in the method *IZ_min*, the four datasets with less accuracy do not drop the accuracies too much ($iris \approx -0.54\%$, $sonar \approx -3.63\%$, *waveform*$\approx -1.94\%$, $wine \approx -2.21\%$). Hence, the Z-value measure provides either a good measurement in the method *IZ* or the*IZ_min*. The experimental results show that the classification results of both the *IZ* and the *IZ_min* are better than the *BZ* and the *BZ_min* obviously. Such results correspond to the discussions of fitness functions in learning process. We also found that both of the conflict rate and the rejection rate of discriminant functions are independent of the number of classification classes but depends on the features in datasets. From Table 11, the vehicle dataset contains four classes with high rejection rate and the bupa dataset has two classes with high conflict rate relatively. However, the

Table 10
The training time of discriminant functions (in s)

Datasets	Classification functions	GP-B				GP-I			
		Average	min	max	*stddev*	*Average*	min	max	*stddev*
bcw	$f_{malignant}$	32.41	23.10	39.31	4.83	15.02	10.90	18.98	2.50
	f_{benign}	33.34	25.61	39.61	4.20	8.39	4.53	15.01	3.31
bupa	f_1	14.78	11.81	18.73	2.32	13.00	7.92	17.67	3.04
	f_2	14.89	11.39	18.60	2.27	13.71	8.80	17.46	2.71
glass	f_{bwfp}	7.37	4.77	13.67	3.33	5.89	3.87	8.31	1.01
	f_{bwnfp}	9.38	5.56	14.63	3.69	7.02	4.54	9.04	1.51
	f_{vwfp}	8.82	4.78	13.51	4.16	6.31	4.19	5.53	1.34
	f_{vwnfp}	8.57	3.31	11.40	2.71	6.18	4.14	5.51	2.07
	$f_{containers}$	7.83	4.44	11.89	3.67	5.21	3.16	9.67	1.74
	$f_{tableware}$	7.62	4.48	11.09	3.10	5.15	3.04	8.19	1.46
	$f_{headlamps}$	7.98	4.18	13.84	4.40	4.92	3.85	9.04	1.73
ionosphere	f_{good}	13.19	9.33	19.57	3.63	10.32	8.10	12.34	1.17
	f_{bad}	10.14	8.76	19.08	4.11	7.51	4.92	10.05	1.57
iris	f_{Setosa}	2.07	1.78	2.35	0.18	1.64	1.31	2.02	0.22
	$f_{Versicolor}$	3.10	2.10	3.73	0.49	2.96	1.86	3.82	0.60
	$f_{Virginica}$	2.40	1.63	3.46	0.57	2.38	1.59	3.55	0.62
pima1	$f_{positive}$	32.51	21.17	42.25	6.40	22.31	10.72	35.30	7.84
	$f_{negative}$	32.93	22.26	49.38	8.91	26.08	12.91	39.59	8.15
pima2	$f_{positive}$	25.87	18.40	33.06	4.54	16.52	5.73	31.71	10.28
	$f_{negative}$	24.66	18.48	31.62	4.24	17.76	6.22	33.95	10.61
sonar	f_M	7.56	4.67	12.18	2.71	5.36	3.68	10.04	1.48
	f_R	7.10	4.05	11.88	3.03	6.13	4.11	9.06	1.77
vehicle	f_{opel}	33.71	25.80	41.31	4.50	30.80	16.75	50.34	9.73
	f_{saab}	34.45	26.18	48.04	6.46	30.21	15.98	41.04	7.77
	f_{van}	34.91	21.16	49.43	8.39	29.59	12.46	48.50	11.87
	f_{bus}	33.61	23.62	48.15	7.48	33.73	17.19	49.17	9.58
waveform	f_{w0}	200.49	181.45	226.30	10.55	157.44	144.75	168.13	6.98
	f_{w1}	197.62	177.59	232.14	12.33	122.73	101.83	143.93	9.74
	f_{w2}	226.11	189.44	240.75	10.09	167.39	136.87	185.80	13.26
wine	f_{wine1}	5.45	4.41	7.99	1.46	4.53	3.67	5.59	0.52
	f_{wine2}	7.30	4.84	10.13	2.62	4.71	3.43	6.11	0.77
	f_{wine3}	6.19	4.71	7.10	0.97	3.85	3.74	4.01	0.08

glass has seven classes but the rates of rejection and conflict are not so high as the vehicle's and the bupa's respectively.

Finally, we compare our results with some well-known previous researches [3,6,22,24,25] in Table 12 . In Ref. [3], Friedman proposed a Bayesian network learning method to build tree augmented Naïve Bayes(TAN) for classification. They also compared the method with Naïve Bayes with 25 datasets via five-fold cross validation. An efficient fuzzy classifier based on fuzzy entropy measure was reported in Ref. [6]. Their experimental results are mainly completed in holdout method and compared with some effective methods. The research of Ref. [22] evaluated and compared thirty-three classification algorithms including twenty-two decision tree, nine statistical and two neural network algorithms in 10-fold cross validation. The other two researches including CBA [24] and SNNB [25] were also done by 10-fold cross validation and made a comparison with the methods of

Table 11
The results of classification

Datasets	Accuracy	GP-B					GP-I				
	(%)	*Acc*	$r_{conflict}$	r_{eject}	*BZ*	*BZ_min*	*Acc*	$r_{conflict}$	r_{reject}	*IZ*	*IZ_min*
bcw	*average*	94.57	1.41	1.82	96.94	94.52	93.06	3.10	1.52	97.17	97.53
	stddev	2.11	1.07	2.02	0.42	1.92	2.58	2.73	0.52	0.45	0.47
bupa	*average*	57.28	11.83	8.61	69.54	69.86	56.38	14.72	8.12	74.70	84.06
	stddev	3.57	4.60	2.34	1.67	2.77	3.02	3.63	1.59	2.05	2.31
glass	*average*	65.23	10.73	4.31	70.08	71.27	70.32	8.57	7.49	75.47	76.81
	stddev	3.06	2.68	2.76	1.53	2.24	2.74	1.08	0.93	0.42	0.94
ionosphere	*average*	77.43	10.24	6.50	88.44	86.11	85.43	7.80	4.91	92.33	94.47
	stddev	1.97	3.26	2.73	2.15	2.62	1.77	0.71	0.88	1.01	2.32
iris (2-fold)	*average*	94.00	3.27	1.60	98.13	93.60	96.53	0.93	1.20	98.27	98.13
	stddev	1.49	1.70	1.53	0.65	2.80	1.45	0.68	1.02	0.53	0.83
iris (10-fold)	*average*	90.53	3.31	4.00	95.67	91.87	91.87	3.20	2.40	96.07	95.53
	stddev	1.83	2.13	1.79	0.95	1.73	1.78	2.15	1.12	1.31	1.19
pima1	*average*	66.34	7.59	6.38	75.03	72.01	66.56	8.14	5.48	76.41	85.31
	stddev	0.94	2.78	1.96	0.99	3.07	1.57	1.80	1.97	0.79	1.95
pima2	*average*	69.06	7.24	5.70	77.54	77.07	56.20	18.12	10.36	79.08	88.01
	stddev	1.68	2.04	1.21	0.73	2.80	2.77	4.57	2.44	1.43	1.64
sonar	*average*	70.51	8.59	4.76	80.98	81.58	80.59	9.06	1.11	88.96	85.33
	stddev	1.84	1.53	2.16	2.39	3.80	1.17	0.99	0.46	0.64	0.88
vehicle	*average*	32.52	13.13	39.09	61.78	59.07	36.82	2.57	48.43	73.72	75.24
	stddev	2.46	4.68	6.14	4.20	2.73	1.85	0.68	2.23	1.25	1.79
waveform	*average*	60.56	14.93	7.73	83.21	82.92	76.46	15.70	6.02	85.50	83.56
	stddev	1.78	2.82	2.11	1.59	2.21	1.09	3.31	1.54	1.14	1.26
wine	*average*	81.72	6.67	4.72	90.91	89.17	90.08	5.66	3.20	95.69	93.48
	stddev	2.75	1.08	1.33	1.12	2.84	1.26	0.96	1.18	0.29	0.44

Table 12
The comparison of the proposed classifiers and the best results summarized in Refs. [3,6,22,24,25]

Datasets	*BZ*	*BZ_min*	*IZ*	*IZ_min*	Previous methods	
bcw	96.94	94.52	97.17	97.53	97.22	(LVQ) [22]
bupa	69.54	69.86	74.70	84.06	72.10	(OCM) [22]
glass	70.08	71.27	75.47	76.81	73.90	(CBA) [24]
ionosphere	88.44	86.11	92.33	94.47	92.10	(CBA) [24]
iris(2-fold)	98.13	93.60	98.27	98.13	97.12	(FEBFC) [6]
iris(10-fold)	95.67	91.87	96.07	95.53	95.30	(C4.5) [24]
pima1	75.03	72.01	76.41	85.31	75.52	(TAN) [3]
pima2	77.54	77.07	79.08	88.01	77.90	(LDA) [22]
sonar	80.98	81.58	88.96	85.33	83.20	(SNNB) [25]
vehicle	61.78	59.07	73.72	75.24	85.50	(QDA) [22]
waveform	83.21	82.92	85.50	83.56	83.90	(NBTree) [25]
wine	90.91	89.17	95.69	93.48	98.30	(NB) [25]

Naïve Bayes and C4.5. Although these researches are done under different environments, the rank of classification rate is consistent. For simplicity, we summarize the classification accuracies from the above researches and list only the best results of different datasets in the last column of Table 12. The classification results of *IZ* and *IZ_min* have the best recognition rates for most of the datasets except the vehicle and the wine. Especially, *IZ_min* improves classification accuracies so much in some medical diagnosis datasets such as the bupa and the pima. For the iris dataset, since the experiment using holdout method is better than 10-fold cross validation, we compare the accuracies of 2- and 10-fold cross validation with the corresponding previous experimental results, respectively. The research results in Ref. [22] show that, for the vehicle dataset, only statistical-based algorithms have higher classification accuracy but it cannot be well classified by other methods. Although the accuracies of *IZ* and *IZ_min* on the vehicle dataset are about ten percent lower than QDA in Ref. [22], they are still better than other methods like NB, C4.5, TAN and CBA. At last, we found that the wine dataset can be well classified by Naïve Bayes-based classifiers. Except Naïve Bayes-based classifiers, *IZ* and *IZ_min* outperform the other methods for the wine dataset.

6. Conclusions

The traditional rule-based classification is to classify patterns using a set of decision rules. For the problem with high-dimensional numerical attributes, a classifier with decision rules may not get a high accuracy of classification and keep rules simply simultaneously. This paper presents a learning approach to generate discriminant functions for classification based on genetic programming. The proposed approaches include an adaptive incremental learning strategy to speed up the training procedure without loss accuracy, a distance-based fitness function to obtain better discriminant function and the mechanism of ambiguity resolution called Z-value measure to resolve not only the cases of conflict but also rejections.

The advantages of classification functions are concise and efficient. The longer training time will produce a classifier with better accuracy without increasing the number of discriminant functions. Hence, the classification rate of a classifier can be easily preserved and improved. However, the main disadvantage of generating a good solution for classification by evolutionary algorithms like genetic algorithm or genetic programming is time consuming. It usually needs much time for the evolution step to learn classification rules or discriminant functions. As we know, the GP learning approaches proposed in Ref. [26] take more than one hour to generate the classification rules on the same datasets we used. However, our experiments show that the proposed GP-B and GP-I learning approaches take only a few seconds or few minutes. They are even faster in comparison with some of the previous methods. We also show that the Z-value measure is effective and the obtained classifiers have high classification rates in comparison with previous methods. Finally, we found that the features selected from the functions can be used to reduce the dimensions of the features of the problem. The feature selection using genetic programming approach thus is an interesting issue for further studying of researchers. Other future extensions on classifying data with symbolic values and missing values using function-based classifier are also worth investigating.

References

[1] J.R. Quinlan, Induction of decision trees, Mach. Learning 1 (1986) 81–106.
[2] J.R. Quinlan, C.45: Programs for Machine Learning, Morgan Kaufmann, Los Altos, CA, 1993.
[3] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (1997) 131–163.

[4] D. Heckerman, M.P. Wellman, Bayesian networks, Comm. ACM 38(3) (1995) 27–30.
[5] R. Kohavi, Scaling up the accuracy of Naïve-Bayes classifiers: a decision tree hybrid, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 202–207.
[6] H.M. Lee, A neural network classifier with disjunctive fuzzy information, Neural Networks 11(6) (1998) 1113–1125.
[7] H.M. Lee, C.M. Chen, J.M. Chen, Y.L. Jou, An efficient fuzzy classifier with feature selection based on fuzzy entropy, IEEE Trans. Systems Man Cybernet. B Cybernet. 31(3) (2001) 426–432.
[8] P.K. Simpson, Fuzzy min–max neural networks—part 1: classification, IEEE Trans. Neural Networks 3 (1992) 776–786.
[9] G.P. Zhang, Neural networks for classification: a survey, IEEE Trans. Systems Man Cybernet. C Appl. Rev. 30(4) (2000) 451–462.
[10] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[11] E.H. Han, G. Karypis, V. Kumar, Text categorization using weight adjusted k-nearest neighbor classification, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2001, pp. 53–65.
[12] C.H. Wang, J.F. Liu, T.P. Hong, S.S. Tseng, A fuzzy inductive learning strategy for modular rules, Fuzzy Set Syst. 103 (1999) 91–105.
[13] C.H. Wang, T.P. Hong, S.S. Tseng, Integrating fuzzy knowledge by genetic algorithms, IEEE Trans. Evolut. Comput. 2(4) (1998) 138–149.
[14] M. Bramrier, W. Banzhaf, A comparison of linear genetic programming and neural networks in medical data mining, IEEE Trans. Evolut. Comput. 5(1) (2001) 17–26.
[15] B.C. Chien, J.Y. Lin, T.P. Hong, Learning discriminant functions with fuzzy attributes for classification using genetic programming, Expert Syst. Appl. 23 (2002) 31–37.
[16] J.K. Kishore, L.M. Patnaik, V. Mani, V.K. Agrawal, Application of genetic programming for multicategory pattern classification, IEEE Trans. Evolut. Comput. 4(3) (2000) 242–258.
[17] J. Sherrah, R.E. Bogner, A. Bouzerdoum, Automatic selection of features for classification using genetic programming, in: Proceedings of the Australian New Zealand Conference On Intelligent Information Systems, 1996, pp. 284–287.
[18] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.
[19] in: J.R. Koza, D.E. Goldberg, D.B. Fogel (Eds.), Genetic Programming, MIT Press, Cambridge, MA, MIT Press, 1996.
[20] C. Blake, E. Keogh, C.J. Merz, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, 1998, http://www.ics.uci.edu/∼mlearn/MLRepository.html.
[21] A. Singleton, Genetic programming with C++, Byte (1994) 171–176.
[22] T.S. Lim, W.Y. Loh, Y.S. Shih, A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms, Mach. Learn. 40 (2000) 203–228.
[23] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, Los Altos, CA, 2001.
[24] B. Liu, W. Hsu, Y. Ma, Integrating classification and association rules mining, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, USA, 1998, pp. 80–86.
[25] Z. Xie, W. Hsu, Z. Liu, M.L. Lee, SNNB: a selective neighborhood based Naïve Bayes for lazy learning, in: Proceedings of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Taiwan, 2002, pp. 104–114.
[26] T. Loveard, V. Ciesielski, Representing classification problem in genetic programming, in: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, pp. 1070–1077.

About the Author—BEEN-CHIAN CHIEN received the Ph.D. in Computer Science and Information Engineering from National Chiao Tung University in 1992. He was an associate professor in Department of Information Engineering in I-Shou University from August 1996 to July 2004. Since August 2004, he is an associate professor in Department of Computer Science and information Engineering in National Tainan Teachers College, Tainan, Taiwan. His current research activities involve machine learning, content-based image retrieval, intelligent information retrieval and data mining.

About the Author —JUNG-YI LIN was born in Taitung, Taiwan. He received the M.S. degree in Computer Science and Information Engineering from I-Shou University in 2002. He is currently a Ph.D. student in Computer and Information Science, National Chiao Tung University, HsinChu, Taiwan. His research interests include machine learning, data mining, and knowledge discovery.

About the Author —WEI-PANG YANG received the Ph.D. degrees in Computer Engineering from the National Chiao Tung University in 1984. Dr. Yang was a visiting scholar at Harvard University and University of Washington at 1986 and 1996, respectively. Currently, he is a professor in Computer and Information Science, and the Director of University Library in National Chiao Tung University, HsinChu, Taiwan. His research interests include database theory, object-oriented database, video database, Chinese database retrieval systems, and digital library.