New C-fuzzy decision tree with classified points

Shiueng-Bien Yang
Leader University
Department of Computer Science and Information Engineering
No. 188 Sec. 5 An-chung Rd.
Tainan City, Taiwan
E-mail: ysb@mail.leader.edu.tw

Abstract. *The C-fuzzy decision tree (CFDT)–based on the fuzzy C-means (FCM) algorithm has been proposed recently. In many experiments, the CFDT performs better than the "standard" decision tree, namely, the C4.5. A new C-fuzzy decision tree (NCFDT) is proposed, and it outperforms the CFDT. Two design issues for NCFDT are as follows. First, the growing method of NCFDT is based on both classification error rate and the average number of comparisons for the decision tree, whereas that of CFDT only addresses classification error rate. Thus, the proposed NCFDT performs better than the CFDT. Next, the classified point replaces the cluster center to classify the input vector in the NCFDT. The classified-points searching algorithm is proposed to search for one classified point in each cluster. The classification error rate of the NCFDT with classified points is smaller than that of CFDT with cluster centers. Furthermore, these classified points can be applied to the CFDT to reduce classification error rate. The performance of NCFDT is compared to CFDT and other methods in experiments.*
 [DOI: 10.1117/1.2976421]

1 Introduction

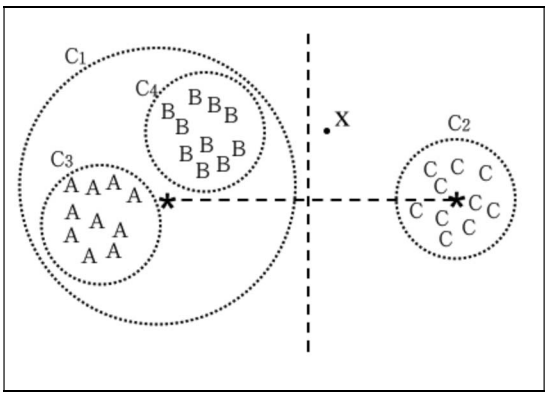
Decision trees[1–10] are common approaches for machine learning, recognition, and classification systems. In the basic design procedure, one variable (attribute) of the input vector is chosen at a time to be compared to the nodes in the decision tree. The users select the most discriminative variable as the new node for growing the decision tree. Cognitive uncertainties, such as fuzziness and ambiguity, have recently been incorporated into the knowledge induction process by using fuzzy decision trees.[11] The Fuzzy ID3 algorithm[12] and its variants[13–15] are popular and efficient methods to design the fuzzy decision trees. Those decision trees can be regarded as single-variable decision trees, in which one variable is considered at a time. This growth method is a drawback when classifying input vectors with multivariable entities characterized by high homogeneity (low variability). In Ref. 16, the data can be perceived as a collection of information granules. Additionally, the information granules and information granulation are almost synonymous with clusters.[17,18] The fuzzy clusters are the central concept behind the generalized tree and are called cluster-oriented decision trees.[19] The C-fuzzy decision tree[19] (CFDT) is derived from the well-known fuzzy C-means algorithm (FCM) clustering methods.[17] The CFDT often outperforms the C4.5.[19] The CFDT is also based on multivariable decision trees. The multivariable decision trees most effectively classify the vectors with multivariable entities characterized by high homogeneity. In contrast to single-variable decision trees, which consider one variable at a time, a multivariable decision tree involves all variables that are considered at each node.

However, the CFDT[19] has two drawbacks. First, the growing method selects the leaf node with the highest classification error rate to be split in the CFDT one at a time. Although the classification error rate can be reduced by splitting a node to grow the decision tree, the number of comparisons is increased when an input vector is classified in the CFDT. However, the average number of comparisons for the input vector classified in the decision tree is not considered during the design of CFDT. The second drawback of CFDT is illustrated in Fig. 1. In Fig. 1(a), the data set consists of three classes, A, B, and C. The data set is divided into two clusters, C_1 and C_2. Cluster C_1 is then further divided into two clusters, C_3 and C_4. Figure 1(b) shows the corresponding CFDT. If input x is classified in the CFDT, input $\mathbf{x}$ is then first compared to clusters C_1 and C_2. Input $\mathbf{x}$ is classified into cluster C_2 because $\mathbf{x}$ is closer to the center of cluster C_2 than the center of cluster C_1. A classification error occurs when class B is the class of input $\mathbf{x}$. This error occurs when a large cluster represents a node. A single center cannot represent all vectors in a large cluster. That is, the center of cluster C_1 is hard to represent all vectors contained in C_1 because C_1 is a large cluster that contains vectors belonging to two classes, A and B. This classification error usually occurs when an input vector is located in a region between the two clusters. Thus, cluster centers are not appropriate for classifying the input vector in the CFDT.

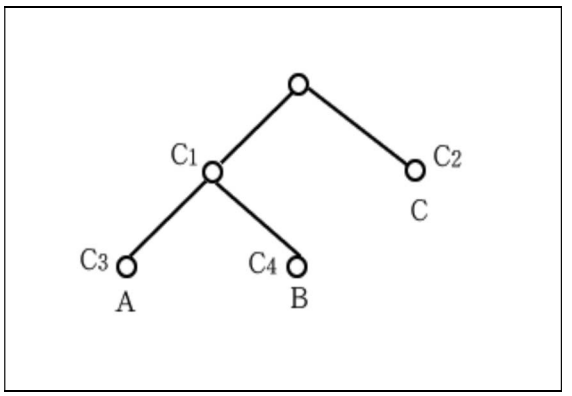
In this paper, a new C-fuzzy decision tree (NCFDT) is proposed that improves these two drawbacks of the CFDT. First, the growing method selects a leaf node to split in the NCFDT, according to classification error rate and the average number of comparisons. Thus, the performance of the NCFDT is better than that of CFDT. The NCFDT then achieves a near-optimal decision tree. Next, the classified points are proposed to replace cluster centers to classify the input vector in the NCFDT. The classified-points searching (CPS) algorithm in the NCFDT is proposed to search for one classified point in each cluster. To reduce the classification error rate in the NCFDT, the classified point in each cluster is closed to the region between clusters. For example, in Fig. 1(c), let P_1 and P_2 be classified points closed

Paper 07239R received Dec. 31, 2007; revised manuscript received Apr. 6, 2008; accepted for publication Apr. 30, 2008; published online Aug. 28, 2008.

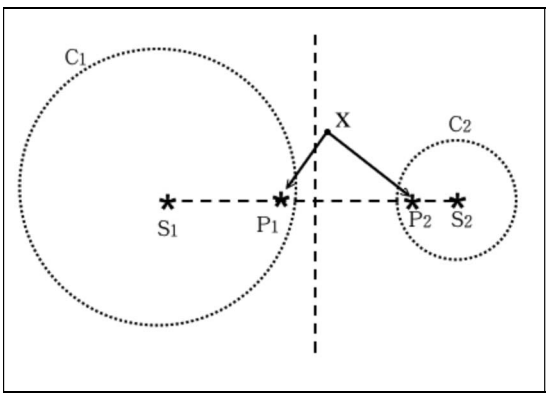
1017-9909/2008/17(3)/033017/11/$25.00 © 2008 SPIE and IS&T.



(a)



(b)



(c)

Fig. 1 Example illustrating the classification error between two clusters: (a) the clustering results, (b) the decision tree, and (c) the input vector $\boldsymbol{x}$ is classified to C_1 because $\boldsymbol{x}$ is closer to P_1 than P_2.

to the region between C_1 and C_2. Input vector $\mathbf{x}$ is closer to P_1 in C_1 than to P_2 in C_2, and, thus, $\mathbf{x}$ is correctly classified to cluster C_1.

The remainder of this paper is organized as follows. Section 2 presents the design of the proposed NCFDT. Section 3 then describes the design of the NCFDT with the classified points. Experiments are given in Section 4, and Section 5 presents conclusions.

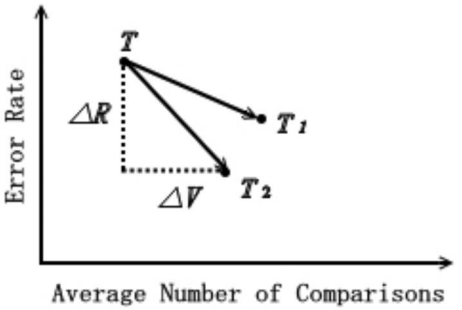


Fig. 2 Example illustrating that T_2 is better than T_1.

2 Design of the NCFDT

The growing method in the CFDT only considered classification error rate of the decision tree when the CFDT was designed. However, classification error rate and the average number of comparisons are both considered when designing the NCFDT. The main goal of NCFDT is to reduce the classification error rate and the average number of comparisons. Figure 2 shows an example to illustrate the growing method. In Fig. 2, T_1 denotes the decision tree after the leaf node t_1 is split in the decision tree T, and T_2 denotes the decision tree after the leaf node t_2 is split in decision tree T. The growing method selects one node at a time to split in T. From Fig. 2, T_2 is better than T_1 because the classification error rate of T_2 is smaller than that of T_1 when the average number of comparisons is the same. In the NCFDT, the growing method selects one node with maximal value of $\lambda = |\Delta R / \Delta V|$ to split in T, while the node with maximal value of ΔR is only considered in the growing method of CFDT. Thus, the performance of the NCFDT is better than that of CFDT.

Before the growing method of NCFDT is described, the average number of comparisons and classification error rate of the NCFDT are defined. First, let T denote the decision tree, NCFDT, and let H denote the set of all leaf nodes in T, including node t_i. Let $P(t_l)$ represent the probability of training vectors in node t_i, which is defined as

$$P(t_i) = \frac{\text{the number of training samples contained in the node } t_i}{\text{all the training samples}}. \tag{1}$$

Let $L(t_i)$ be the average number of comparisons required when input $\mathbf{x}$ travels from the root node to the leaf node t_i in the NCFDT. Then, the average number of comparisons of T, $V(T)$, is defined as

$$V(T) = \sum_{t_i \in H} P(t_i) L(t_i) \tag{2}$$

The classification error rate of the NCFDT is then defined as follows. Let $X(k) \in R^n$ be the training vector, and let $Y(k) \in R$ be the corresponding class of $X(k)$. Let cluster C_i be the set of training vectors contained in node t_i. Then,

$$C_i = \{X(k) | u_i[X(k)] > u_j[X(k)], \quad \text{for all } j \neq i\}, \tag{3}$$

where $u_i[X(k)]$ denotes the degree of membership that $X(k)$ has in cluster C_i. Let there be z leaf nodes, including node t_i that have the same parent node. Then,

$$u_i[X(k)] = \frac{1}{\sum_{j=1}^{z} [\|X(k) - S_i\| / \|X(k) - S_j\|]^2}, \tag{4}$$

where $\| \|$ indicates the measure of Euclidean distance and S_i is the center of C_i. Notably, $u_i[X(k)]$ satisfies the constraints that $0 \leqslant u_i[X(k)] \leqslant 1$ and $\sum_{i=1}^{z} u_i[X(k)] = 1$. Furthermore, after training vectors are classified to clusters, the new center S_i' of cluster C_i is updated as

$$S_i' = \frac{\sum_{X(k) \in C_i} u_i[X(k)]^2 X(k)}{\sum_{X(k) \in C_i} u_i[X(k)]^2}. \tag{5}$$

Let the set O_i collect outputs of training vectors that have been assigned to C_i as

$$O_i = \{Y(k) | X(k) \in C_i\}. \tag{6}$$

The definition of the representative of t_i, positioned in the output space, is given as

$$M_i = \frac{\sum_{[X(k),Y(k)] \in C_i \times O_i} u_i[X(k)] Y(k)}{\sum_{[X(k),Y(k)] \in C_i \times O_i} u_i[X(k)]}. \tag{7}$$

Then, the classification error rate of node t_i, $R(t_i)$, is then defined as

$$R(t_i) = \sum_{[X(k),Y(k)] \in C_i \times O_i} u_i[X(k)][Y(k) - M_i]^2. \tag{8}$$

When $R(t_i) = 0$, the node t_i can be assigned to a pure class, and then node t_i will not be further divided while growing the decision tree. Furthermore, when $R(t_i) > 0$, training vectors contained in the data set of node t_i still belong to many classes, and node t_i is hard to be positioned in the output space. In this case, node t_i can be further split to grow the decision tree. Finally, the classification error rate of T, $R(T)$, is defined as

$$R(T) = \sum_{i \in H} P(t) R(t). \tag{9}$$

Let tree T' indicate that tree T after node t_i is split into z child nodes, $t_{i,1}, t_{i,2}, \ldots, t_{i,z}$, by the FCM. Then,

$$V(T) = \sum_{\substack{j \in H \\ j \neq t_i}} P(j) Y(j) + P(t_i) V(t_i), \tag{10}$$

$$R(T) = \sum_{\substack{j \in H \\ j \neq t_i}} P(j) R(j) + P(t_i) R(t_i), \tag{11}$$

$$V(T') = \sum_{\substack{j \in H \\ j \neq t_i}} P(j) V(j) + \sum_{k=1}^{z} P(t_{i,k}) V(t_{i,k}) \tag{12}$$

$$R(T') = \sum_{\substack{j \in H \\ j \neq t_i}} P(j) R(j) + \sum_{k=1}^{z} P(t_{i,k}) R(t_{i,k}). \tag{13}$$

Under the conditions of quasi-convexity, the slope of the classification error rate and the average number of comparisons between that of T and T' is

$$\lambda = \frac{\Delta R}{\Delta V} = \frac{R(T) - R(T')}{V(T') - V(T)} = \frac{P(t_i) R(t_i) - \sum_{k=1}^{z} P(t_{i,k}) R(t_{i,k})}{[\sum_{k=1}^{z} P(t_{i,k}) V(t_{i,k})] - P(t_i) V(t_i)}. \tag{14}$$

Thus, the aim is to maximize λ when splitting a node in the decision tree to reduce classification error rate and the average number of comparisons. The growing method selects the node with the largest λ to be split at a time in the NCFDT.

In the paper, the design issue of the proposed NCFDT focuses on the storage constraint condition. Before designing the NCFDT with the storage constraint, the user must give a threshold, δ, as the storage constraint. The NCFDT with this storage constraint is grown until the storage of NCFDT reaches the threshold, δ. The algorithm for designing the NCFDT with storage constraint is described as follows.

Algorithm: Design_NCFDT_Storage_Constraint

Input: Storage threshold δ and the training data set.

Output: The NCFDT with the storage constraint.

Step 1. Let the root node of tree T contain all training vectors in the training data set. Set $STORAGE = 0$.

Step 2. **While** $STORAGE < \delta$

Step 2.1. For each leaf node t and T, such that $R(t) > 0$, perform the following.

Step 2.1.1. Apply the FCM to all training vectors in node t to generate z child nodes of t.

Step 2.1.2. Calculate the value of λ while node t is split in T.

Step 2.2. Let t' be the leaf node with the largest λ, and let t' be divided into z child nodes by the FCM. Split node t' in T to generate the new decision tree T'.

Step 2.3. $STORAGE = STORAGE + z$ and set $T = T'$.

Step 3. Output the NCFDT with storage constraint, T.

3 Design of the NCFDT with Classified Points

Section 3.1 analyzes the classified points. Section 3.2 describes the method for designing classified points in a cluster using the CPS algorithm. Section 3.3 describes the classification method for the NCFDT with classified points.

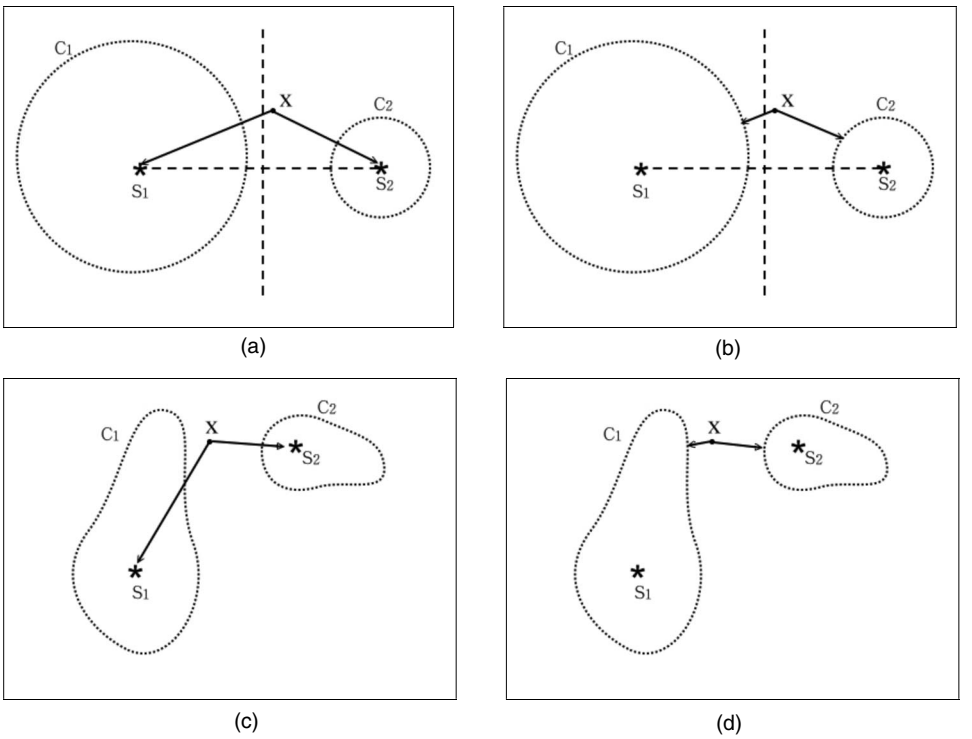


Fig. 3 Comparisons of two distance measures, *Center_Dist* and *Vector_Dist*: (a) and (c) $Center\_Dist(x, C_1) > Center\_Dist(x, C_2)$, and (b) and (d) $Center\_Dist(x, C_1) < Vector\_Dist(x, C_2)$.

3.1 *Analysis of Classified Points*

The cluster center is defined as the mean of the vectors contained in the cluster, and then the cluster center represents all of the vectors contained in the cluster. Thus, the distance between the input vector and the cluster center is usually used to measure the distance between the input vector and the cluster. Let cluster C_i contain m_i vectors, $\mathbf{X}^i(1), \mathbf{X}^i(2), \ldots, \mathbf{X}^i(m_i)$, and let $\mathbf{x}$ be an input vector. The distance between $\mathbf{x}$ and C_i, $Center\_Dist(\mathbf{x}, C_i)$, is then defined as

$$Center\_Dist(\mathbf{x}, C_i) = \|\mathbf{x} - S_i\|, \tag{15}$$

where S_i denotes the cluster center of C_i. Furthermore, the cluster center is popularly used to classify the input vector in the decision tree, such as the CFDT. However, two cases exist in which the cluster center cannot classify the input vectors well in the decision tree. Figure 1 shows one such case, in which the vectors are variant in the large cluster, meaning that the single cluster center cannot adequately represent all the vectors in the large cluster. In the other case, the cluster center cannot significantly represent all the vectors contained in the cluster with nonspherical shape and therefore cannot classify the input vector well in a decision tree designed with nonspherical clusters. These above two cases indicate that the cluster center is not the best approach for classifying input vectors in a decision tree.

Before designing the classified point of the cluster, the distance measure between the input vector $\mathbf{x}$ and the cluster C_i, $Vector\_Dist(\mathbf{x}, C_i)$, is defined as follows:

$$Vector\_Dist(\mathbf{x}, C_i) = \min_{1 \le k \le m_i} \|\mathbf{x} - X^i(k)\|. \tag{16}$$

Figure 3 shows two examples, the large cluster and the clusters with nonspherical shape, to compare two distance measures, *Vector_Dist* and *Center_Dist*. In Figs. 3(a) and 3(c), the input vector $\mathbf{x}$ is wrongly classified to C_2 by the *Center_Dist* measure, but is accurately classified to C_1 by the *Vector_Dist* measure as in Figs. 3(b) and 3(d). From Fig. 3, the *Vector_Dist* is better than the *Center_Dist* for classifying the input vectors between the two clusters. However, the *Vector_Dist* has a higher time complexity than the *Center_Dist*. The time complexities of *Vector_Dist* and *Center_Dist* are $O(m_i)$ and $O(1)$, respectively.

To reduce the time complexity of the *Vector_Dist* measure, Eq. (16) should be simplified. The nearest points of cluster C_i are first defined as follows. The nearest points in C_i are vectors closest to other clusters. Let $\mathbf{X}^j(\mathbf{h}) \in C_j$ be one of the vectors in C_j. The Euclidean distances between $X^j(h)$ and vectors contained in cluster C_i are calculated. Let $\mathbf{X}^i(\mathbf{r})$ satisfy

$$\|\mathbf{X}^j(\mathbf{h}) - \mathbf{X}^i(\mathbf{r})\| = \min_{x'(k) \in C_i} \|X^j(h) - X^i(k)\|, \quad i \neq j. \tag{17}$$

The vector $\mathbf{X}^i(\mathbf{r})$ can then be relabeled as $\hat{\mathbf{X}}^i(\mathbf{r})$, which is regarded as the nearest point of cluster C_i. Let the cluster C_i contain f_i nearest points, $\hat{\mathbf{X}}^i(1), \hat{\mathbf{X}}^i(2), \ldots, \hat{\mathbf{X}}^i(f_i)$, and let the input vector $\mathbf{x}$ belong to C_j. Equation (16) can be simplified as

$$Vector\_Dist(x,C_i) = \min_{1\leqslant k\leqslant m_i} \|\mathbf{x}-\mathbf{X}^i(k)\| = \min_{1\leqslant l\leqslant f_i} \|\mathbf{x}-\hat{\mathbf{X}}^i(l)\|. \tag{18}$$

The time complexity of $Vector\_Dist(x,C_i)$ is reduced to $O(f_i)$. Notably, $f_i<m_i$.

This paper proposes the classified point to replace the cluster center to classify the input vectors in the decision tree. The classified point of the cluster is defined as follows. Let T be the NCFDT. Except for the root node in T, each node contains a classified point in the NCFDT. Let z nodes, $t_1,t_2,\ldots,t_z$, have the same parent node in T, and let these z child nodes be represented by z clusters, $C_1,C_2,\ldots,C_z$. Let cluster C_i contain m_i vectors, $\mathbf{X}^i(1),\mathbf{X}^i(2),\ldots,\mathbf{X}^i(m_i)$. Let P_i be the classified point in the cluster C_i, for $1\leqslant i\leqslant z$. The classified point P_i in the cluster C_i thus satisfies

$$\|\mathbf{X}^i(l)=P_i\| < \min_{\substack{1\leqslant j\leqslant z \\ i\neq j}} \|\mathbf{X}^i(l)-P_j\| \tag{19}$$

where $\mathbf{X}^i(l)\in C_i$, $1\leqslant l\leqslant m_i$. If classified point P_i satisfies Eq. (19), then P_i can replace the cluster center to represent all vectors contained in cluster C_i.

Let P_i be the classified point of cluster C_i. The distance measure, $CP\_Dist(\mathbf{x},C_i)$, is defined to calculate the distance between the input vector $\mathbf{x}$ and the cluster C_i with the classified point. Then,

$$CP\_Dist(x,C_i) = \|x-P_i\|. \tag{20}$$

The time complexity of $CP\_Dist(\mathbf{x},C_i)$ is $O(1)$ as that of $Center\_Dist(\mathbf{x},C_i)$. If the classified point P_i approaches the nearest points of cluster C_i, then the classified point P_i in C_i is also close to the other clusters. The following section describes how to find the classified point in a cluster by the proposed CPS algorithm.

3.2 *Design of CPS*

In traditional decision trees, including the CFDT, each node can be represented by a cluster center. Therefore, the input vector traces the decision tree from the root node to a leaf node by comparing the input vector to the cluster center included in the node. However, in the NCFDT, the classified point instead of the cluster center is used for comparison to the input vector. The CPS is proposed to search for the classified point in each cluster; this process is as follows. There are two stages to design the CPS. In the first stage, many nearest points are selected from each cluster. In general, these nearest points in the cluster are also the vectors closest to other clusters. The second stage of the CPS is a genetic algorithm that searches for the classified point of each cluster, and the classified point is close to these nearest points in each cluster.

The algorithm of the first stage of CPS, namely CPS-(first stage), is described for searching the sets of nearest points in the cluster.

Algorithm: CPS(first stage)

Input: Let z nodes, $t_1,t_2,\ldots,t_z$, have the same parent node in the decision tree, and let these z nodes be represented by z clusters, $C_1,C_2,\ldots,C_z$.

Output: Sets of nearest point of cluster C_i, $\phi(C_i)$, for $1\leqslant i\leqslant z$.

Step 1. For each cluster C_i, for $1\leqslant i\leqslant z$, do the following.

Step 1.1. Set $\phi(C_i)$ to empty.

Step 1.2. For each vector, $X^j(h)\in C_j$, where $i\neq j$, do the following.

Calculate the Euclidean distances between $\mathbf{X}^j(\mathbf{h})$ and vectors contained in cluster C_i. Let $\mathbf{X}^i(\mathbf{r})$ satisfy Eq. (17). If $\mathbf{X}^i(\mathbf{r})$ is not in the set, $\phi(C_i)$, $\mathbf{X}^i(\mathbf{r})$ is regarded as a nearest point in cluster C_i, and $\mathbf{X}^i(\mathbf{r})$ is added to set $\phi(C_i)$.

Step 2. Output sets, $\phi(C_i)$, for $1\leqslant i\leqslant z$. End.

Figure 4 shows an example illustrating the nearest points. Let the vectors in node t of the decision tree be clustered into three clusters, C_1, C_2, and C_3, using the FCM [Fig. 4(a)]. In Fig. 4(b), each vector in a solid-line circle represents the nearest point.

The second stage of the CPS, namely, CPS(second stage), is a genetic algorithm that searches for classified point P_i in cluster C_i using set $\phi(C_i)$. The genetic algorithm consists of an initialization step and many generations. Each generation consists of three phases, namely, reproduction, crossover, and mutation. The following describes how to search for classified point P_i in cluster C_i by the CPS-(second stage).

3.2.1 *Initialization step*

Let $S_i=(S_{i,1},S_{i,2},\ldots,S_{i,n})\in R^n$ and $P_i=(P_{i,1},P_{i,2},\ldots,P_{i,n})\in R^n$ be two n-dimensional vectors that are the center and classified point in cluster C_i, respectively. In the initialization step, a population of N strings is randomly generated randomly. Each string contains n values, $P_{i,1},P_{i,2},\ldots,P_{i,n}$, that represent these n features of P_i. Let $G=(G_1,G_2,\ldots,G_n)$ be a string in the population. Then,

$$G_k = S_{i,k}+\varepsilon_k, \quad \text{for } 1\leqslant k\leqslant n, \tag{21}$$

where ε_k is generated randomly. Each string represents a solution for P_i.

3.2.2 *Reproduction phase*

If string G is a good solution to represent the classified point P_i of cluster C_i, then each vector $\mathbf{X}^i(l)\in C_i$ satisfies the follow:

$$\|\mathbf{X}^i(l)-P_i\| < \min_{\substack{1\leqslant j\leqslant z \\ i\neq j}} \|\mathbf{X}^l(l)-\mathbf{X}^j(k)\|. \tag{22}$$

Let δ_G be the number of vectors that satisfy Eq. (22) in cluster C_i. Then,

$$\delta_G = \sum_{k=1}^{m_i} \delta_k, \tag{23}$$

where, if vector $\mathbf{X}^i(l)\in C_i$ satisfies Eq. (22), δ_k is set to 1; otherwise, δ_k is set to 0. The first design issue concerning the fitness function emphasizes that all vectors in a cluster should be closer to its classified point rather than other clusters. Thus, δ_G must be as large as possible. If $\delta_G=m_i$,

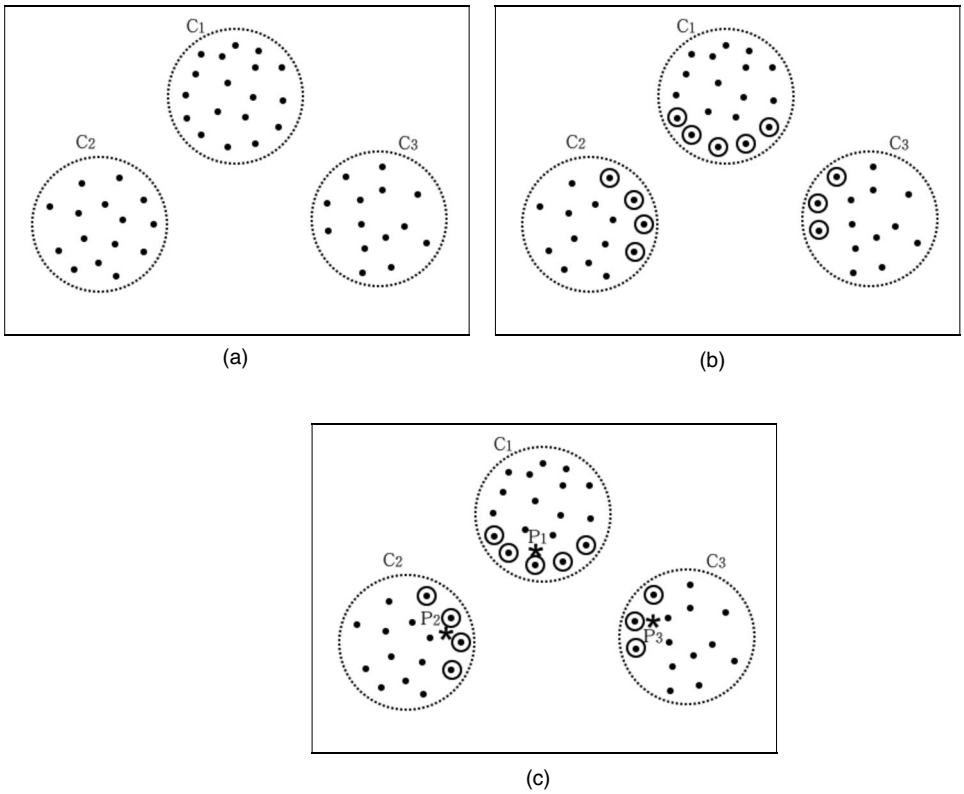


Fig. 4 Example illustrating the classified points in three clusters: (a) the vectors in three clusters, (b) the nearest points in three clusters, and (c) the classified points in three clusters.

classified point P_i can represent all vectors in cluster C_i.

The second design issue of the fitness function emphasizes that classified point P_i should be close to these nearest points in $\phi(C_i)$. The classified point is required to be close to the nearest points in a cluster to reduce classification error in regions between clusters. Let $\phi(C_i) = \{\hat{\mathbf{X}}^i(1), \hat{\mathbf{X}}^i(2), \ldots, \hat{\mathbf{X}}^i(f_i)\}$ consist of f_i nearest points in cluster C_i. Then, Ω_G is defined as

$$\Omega_G = \frac{\Sigma_{k=1}^{f_i} \| P_i - \hat{X}^i(k) \|}{f_i}, \tag{24}$$

which is emphasized as small as possible in the fitness function.

To summarize these two design issues, the fitness function for string G is defined as

$$Fitness(G) = \begin{cases} \dfrac{\delta_G}{m_i} & \text{if } \delta_G < m_i \\ 1 + \dfrac{\theta}{\Omega_G} & \text{if } \delta_G = m_i \end{cases}, \tag{25}$$

where θ is a constant. In Eq. (25), if $\delta_G < m_i$, the classified point P_i cannot represent all vectors in cluster C_i, and $Fitness(G)$ is then set to a small value within $[0, 1]$; otherwise, $Fitness(G)$ exceeds 1. After the fitness of all strings in the population is calculated, the reproduction operator is implemented using a roulette wheel with slots sized according to fitness.

3.2.3 *Crossover and mutation phases*

Two random numbers in $[1, n]$ are generated to determine which values of two strings are to be interchanged in the crossover phase. Furthermore, in the mutation phase, the value of each string G in the population are chosen from $[1, n]$ with a probability to do the mutation. Each chosen value is added to a constant generated randomly.

After the genetic algorithm is applied to cluster C_i, the string $\hat{G}$ with the best fitness, which represents the best solution for P_i, is reserved. Figure 4(c) shows an example that illustrates the classified points in clusters.

The CPS algorithm is used to find the classified point in a cluster whose time-complexity is analyzed as follows. Let there be z clusters, and the total number of all vectors contained in these z clusters be m. Let the average number of vectors contained in C_i be m_i. The time complexity of the CPS(first stage) is dominated by step 1, which takes $\mathbf{O}(mm_i)$ time to calculate the Euclidean distances between pairs of vectors when determining the nearest points of cluster C_i. In the CPS(second stage), time complexity is dominated by the calculation of the fitness function. It takes $\mathbf{O}(mm_i)$ time to calculate the value of δ_G. Let N denote population size. Thus, the reproduction phase takes $\mathbf{O}(Nmm_i)$ time in the worst case. Suppose that the genetic algorithm runs Q generations, time complexity will be $\mathbf{O}(QNmm_i)$. Hence, the time complexity of the whole algorithm is $\mathbf{O}(QNmm_i)$.

Figure 5 illustrates the design of NCFDT with classified points. Figure 5(a) shows the training data set with 50 train-

ing vectors, which can be classified into two classes, with 25 training vector for each class. Figure 5(b) shows the clustering results when the NCFDT is designed, and Fig. 5(c) displays the corresponding decision tree of NCFDT. Table 1 lists the cluster centers and classified points contained in the nodes of the NCFDT tree.

3.3 *Classification Method of the NCFDT with Classified Points*

Before describing the classification method for the NCFDT with classified points, the output class represented by each leaf node is defined. Let the NCFDT T be designed for handling the M-classes problem. Let leaf node t in T contain m_i vectors, $\mathbf{X}^i(k)$, for $1 \leqslant k \leqslant m_t$, and let the corresponding output of class $\mathbf{X}^i(k)$ denote $\mathbf{Y}^i(k)$. Then, $Output(j)$, for $1 \leqslant j \leqslant M$, is calculated as

$$Output(j) = \sum_{k=1}^{m_i} u_t[X^t(k)]\alpha^j(k), \tag{26}$$

where $u_t[\mathbf{X}^t(k)]$ denotes the degree of membership such that $\mathbf{X}^t(k)$ is in node t, which is defined in Eq. (4), and

$$\alpha^j(k) = \begin{cases} 1 & \text{if } Y^i(k) = j \\ 0 & \text{otherwise} \end{cases}. \tag{27}$$

Then, output class w_t, assigned to the leaf node t, is defined as

$$w_t = \arg \max_{1 \leqslant j \leqslant M} Output(j). \tag{28}$$

The classification method of NCFDT with classified points is as follows:

Algorithm: Classification_NCFDT_Classified_Points

Input: Input vector, **x**. The NCFDT T, in which each node contains a classified point.

Output: The output of class.

Step 1. Let t be the root node of T.

Step 2. **While** t is not a leaf node in T

Step 2.1. Let node t contain z child nodes, $t_1, t_2, \ldots, t_z$, and let each node t_i contain a singleclassified point, P_i. Calculate the $u_i(x)$ of node t_i, as follows.

$$u_i(x) = \frac{1}{\sum_{j=1}^{z}[\|x - P_i\|/\|x - P_j\|]} \quad \text{for } 1 \leqslant i \leqslant z.$$

Step 2.2. Let $k = \arg \max_{1 \leqslant i \leqslant z} u_i(x)$. Set $t = t_k$.

Step 3. Output the class w_t assigned to the leaf node t.

The time complexity of the classification algorithm of the NCFDT is as follows. Suppose that the total number of leaf nodes in the NCFDT is n; the average number of child nodes of each internal node in the NCFDT is z. Then, time complexity is approximately $\mathbf{O}\{z[\log_z(n)]\}$ for the classification algorithm of the NCFDT.

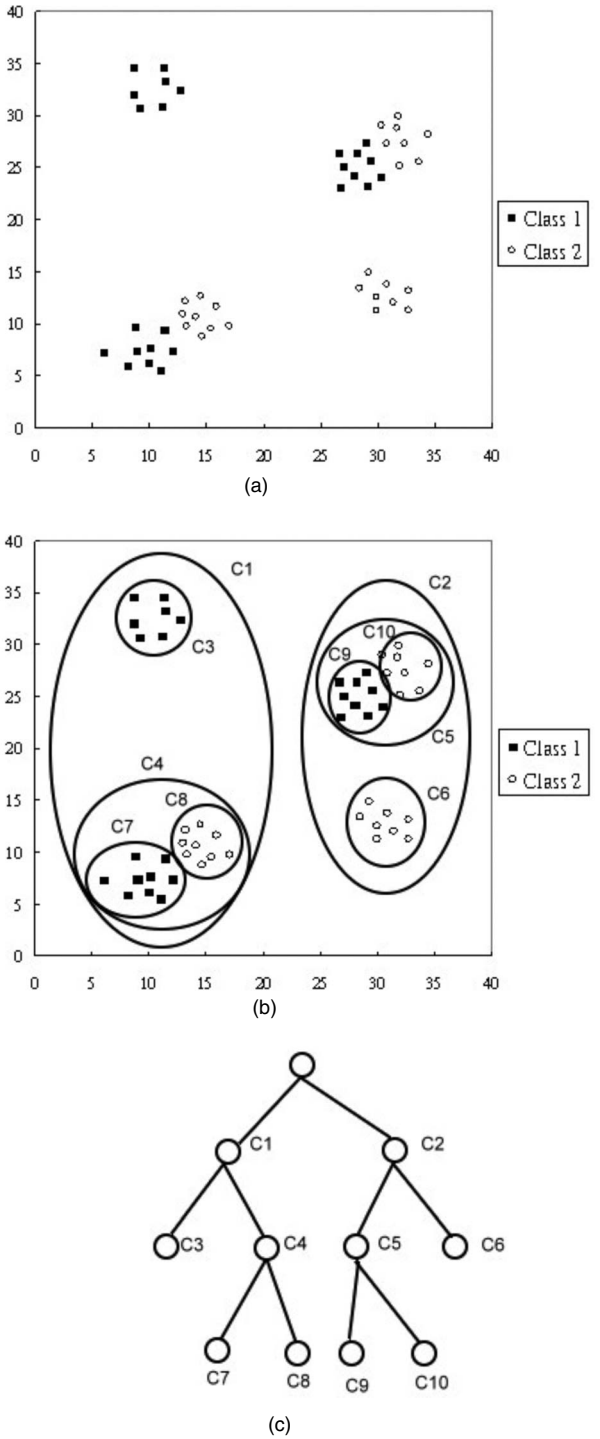


Fig. 5 Example to illustrate the decision tree of NCFDT: (a) two-dimensional training data, (b) the clustering results of NCFDT, and (c) the decision tree of NCFDT.

4 Experiments

Ten data sets were used to test the proposed NCFDT and four established decision trees, namely, CFDT, MB,[14] Fuzzy-ID3(I),[12] and Fuzzy-ID3(II).[15] All these decision trees can be classified as single variable and multivariable, according to the design of conception. NCFDT and CFDT are multivariable decision trees, and the other three methods, MB, Fuzzy-ID3(I), and Fuzzy-ID3(II), are single-

Table 1 The *x*- and *y*-axis of the cluster centers and classified points contained in the nodes of the NCFDT tree.

Clusters	Cluster Centers	Classified Points
C1	(10.21, 15.35)	(18.35, 19.93)
C2	(30.54, 23.47)	(24.45, 22.37)
C3	(10.33, 35.65)	(11.62, 28.32)
C4	(12.54, 10.53)	(13.43, 12.37)
C5	(29.42, 26.74)	(32.65, 21.76)
C6	(30.37, 11.39)	(31.02, 15.32)
C7	(10.43, 6.47)	(12.56, 9.54)
C8	(14.54, 12.63)	(13.03, 9.98)
C9	(27.43, 25.76)	(30.21, 26.33)
C10	(31.65, 26.55)	(31.87, 26.43)

Table 2 Description of data sets.

Data Set	No. Features	No. Vectors	No. Classes
Glass	9	214	7
Hepatitis	19	155	2
Ionosphere	34	351	2
Iris	4	150	3
Pima	8	768	2
Abalone	8	4177	29
Pendigits	16	10992	10
Letter	16	20000	26
Texture	8	30720	30
Speech	480	6240	26

variable decision trees. The multivariable decision trees most effectively classify the vectors with multivariable entities characterized by high homogeneity (low variability). In contrast with single-variable decision trees, which consider one variable (feature) at a time, a multivariable decision trees involves all variables that are considered at each node.

Section 4.1 describes the ten data sets used in the experiments. Section 4.2 shows the performance of comparison between NCFDT and other methods.

4.1 *Data Sets*

Ten data sets, comprising eight from the UC Irvine repository,[20] the texture data set shown in Fig. 6, and the speech data set extracted from the ISOLET database, were adopted in this experiment. Table 2 lists the features of ten data sets. In the texture data set, 30 texture images in Ref. 21 (Fig. 6), were used to test the NCFDT. Each texture (512×512 pixels with 256 gray levels) was divided into blocks with size 16×16 pixels. Each block was then transformed by a Haar wavelet transform[22] to obtain four subbands. The mean values (mv) and standard deviations (sd) of the four subbands were calculated as follows:

$$mv = \frac{1}{N^2}\sum_{i,j=1}^{N} v(i,j), \tag{29}$$

$$sd = \sqrt{\frac{1}{N^2}\sum_{i,j=1}^{N}[v(i,j)-mv]^2}, \tag{30}$$

where N denotes the size of the subband (set to 8 in this experiment) and $v(i,j)$ indicates the wavelet coefficient at location (i,j) in the subband. Therefore, each block containing four subbands can be represented by a feature vector with eight values, because two values, sd and mv, exist for each subband.

In the speech data set, the ISOLET database using the 26 letters of the English alphabet was used in the isolated word recognition test. The speech data set consisted of 6240 utterances from 120 speakers. Each utterance was sampled at 16 kHz with a 16-bit resolution. A Hamming window of 20 ms with 50% overlap was used to process each utterance further by fast Fourier transform (FFT). Each utterance was divided into 15 Hamming windows, each represented by 32 FFT coefficients. That is, each utterance consisted of 480 features.

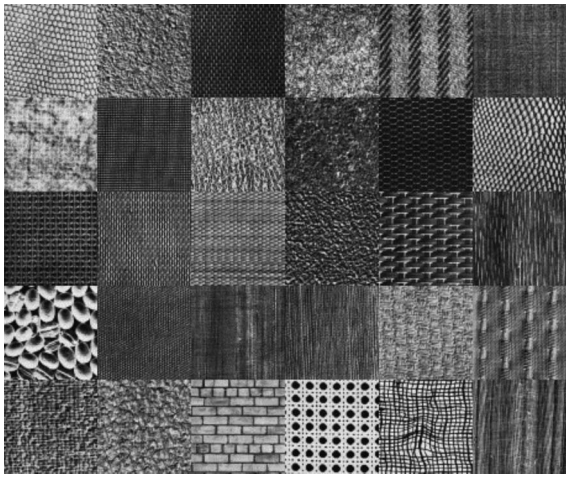


Fig. 6 Thirty textures from the Brodatz album. Row 1: D3, D4, D6, D9, D11, D16. Row 2: D19, D21, D24, D29, D34, D36. Row 3: D52, D53, D55, D57, D65, D68. Row 4: D74, D77, D78, D79, D82, D83. Row 5: D84, D92, D95, D102, D103, D105.

4.2 *Performance of NCFDT*

A stratified tenfold cross-validation was performed for each data set. The stratified tenfold cross-validation breaks the data set into ten disjoint subsets, each with a class distribu-

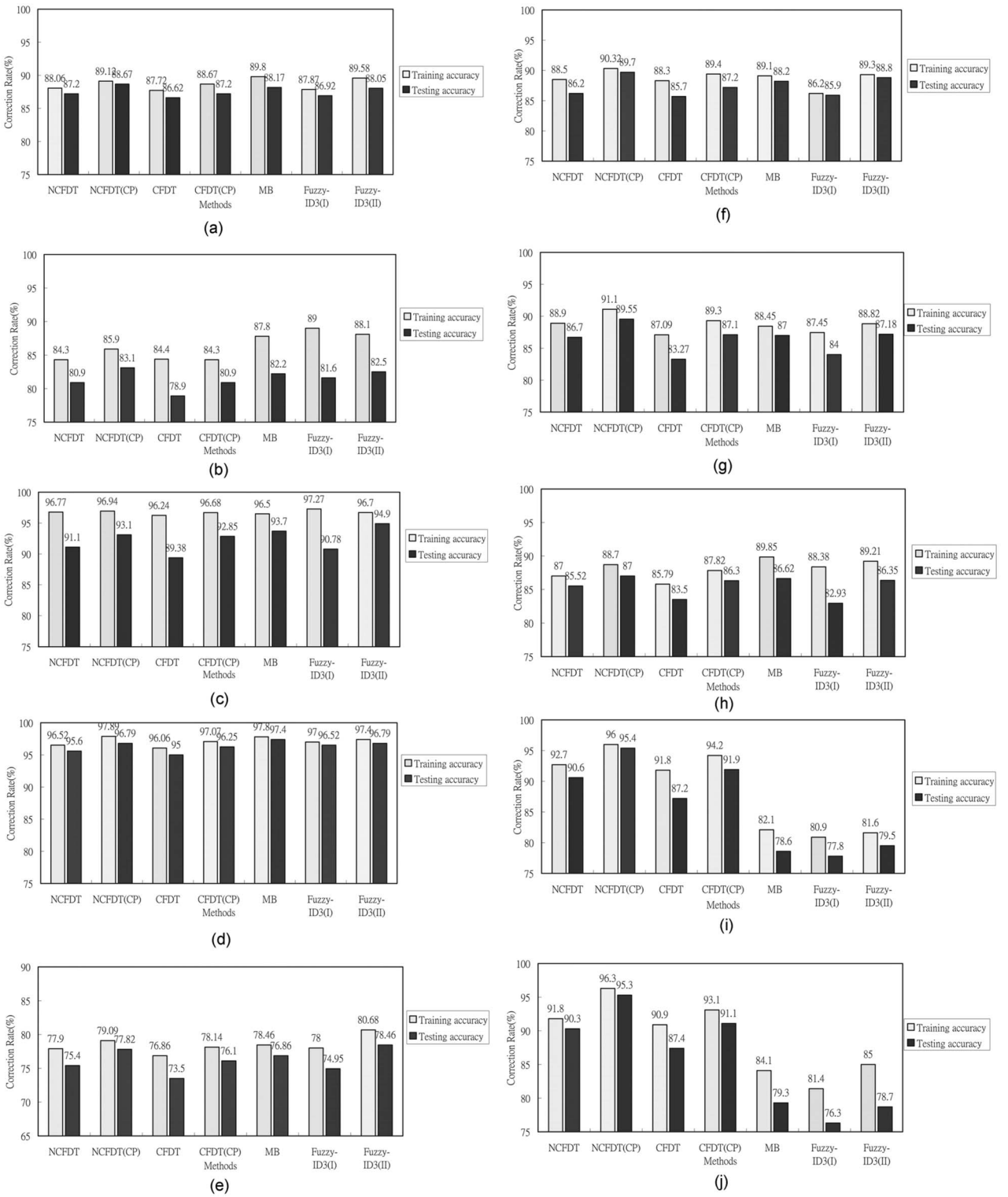


Fig. 7 Comparative analysis for several methods: (a) glass data set, (b) hepatitis, (c) ionosphere, (d) iris, (e) pima, (f) abalone, (g) pendigitis, (h) letter, (i) texture, and (j) speech data sets.

tion approximating that of the original data set. For each of the tenfolds, an ensemble was trained using nine of subsets and tested on the held out subset. The testing accuracy of a decision tree thus denotes the average recognition rate, tenfold. Furthermore, the CPS algorithm was adopted to search for the classified point of each node in NCFDT(CP). The time complexity of CPS is dominated by the second stage of CPS, called CPS(second stage). In the CPS(second stage), the population size was set to 100, and the crossover and mutation probabilities were set to 80 and 5%, respec-

Table 3 Number of nodes used to design the NCFDT and CFDT.

Data Set	No. Nodes
Glass	20
Hepatitis	15
Ionosphere	25
Iris	10
Pima	30
Abalone	60
Pendigits	50
Letter	60
Texture	60
Speech	80

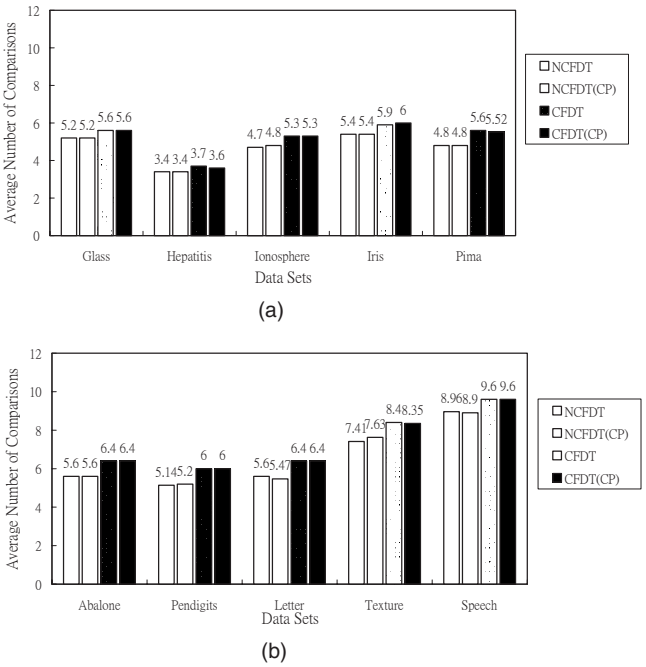


Fig. 8 Average number of comparisons of NCFDT and CFDT.

tively. The genetic algorithm in CPS(second stage) was run over 300 generations, and the best solution obtained from these 300 generations was retained. The users were normally satisfied with the efficiency of CPS(second stage).

The NCFDT was compared to the other single-variable and multivariable decision trees. First, NCFDT was compared to the multivariable decision tree, CFDT. To compare these two methods fairly, the same number of nodes (storage) was applied to design the decision trees for each data set. Table 3 lists the number of nodes used in designing the NCFDT and CFDT for each of the ten data sets. Figure 7 shows the recognition rates of NCFDT and CFDT. In Fig. 7, NCFDT denotes that the cluster center in each node was used to classify the input vector, and NCFDT(CP) indicates that the classified point contained in each node was used to classify the input vector. Additionally, the classified points can be applied to the design of CFDT, which is denoted as CFDT(CP). The following two conclusions can be obtained from Fig. 7: (i) NCFDT(CP) and CFDT(CP) outperformed NCFDT and CFDT, respectively, for each of the ten data sets. This is because the classified point in each cluster was close to the other clusters, and the classification error in the regions between any two pairs of clusters could thus be reduced in the decision tree. (ii) NCFDT(CP) and NCFDT outperformed CFDT(CP) and CFDT, respectively, for each of the ten data sets. This is because the growth method of NCFDT is based on the classification error rate and the average number of comparisons of the decision tree, whereas that of CFDT only considers the classification error rate. In the NCFDT, the decision tree is grown by selecting the node with the largest value of λ to be split. Figure 8 also shows that the average numbers of comparisons in NCFDT and NCFDT(CP) are smaller than those in CFDT and CFDT(CP), respectively. These two findings reveal that NCFDT outperformed CFDT.

NCFDT was then compared to three single-variable decision trees, MB, Fuzzy-ID3(I), and Fuzzy-ID3(II). Figures 7(a)–7(h) reveal that the users were not easily able to determine which one of these four methods, NCFDT(CP), MB, Fuzzy-ID3(I), and Fuzzy-ID3(II), was the best decision tree. However, NCFDT(CP) and CFDT(CP) were observed to outperform MB, Fuzzy-ID3(I), and Fuzzy-ID3(II), from Figs. 7(i) and 7(j). The reason is that the variables in the vectors had high homogeneity for texture and speech data sets. That is, vectors with high homogeneity could not be classified in single-variable decision trees, which consider one variable at a time. The multivariable decision trees, namely, NCFDT(CP) and CFDT(CP), classified the vectors contained in the texture and speech data sets more effectively than the single-variable decision trees. However, one disadvantage of the multivariable decision trees is that they took a long time to classify each input vector. In contrast to single-variable decision trees, in which one variable is considered at a time, a multivariable decision tree considers all variables at each node of the tree.

5 Conclusion

The NCFDT is an efficient decision tree that has been applied to many classification applications. The growing method of the NCFDT selects one node to split according to the classification error rate and the average number of comparisons of the decision tree, whereas the growing method of the CFDT only considers classification error rate. In experiments, the NCFDT performed better than the CFDT. Furthermore, the classified point replaces the cluster center when classifying the input vector in the decision tree. In this paper, the CPS algorithm is proposed to search for the classified point in a cluster. To reduce further the classification error rate in the NCFDT, the CPS algorithm emphasizes that the classified point in a cluster is close to other clusters. In the experiments, the NCFDT with classified points has a smaller classification error rate than the NCFDT with cluster centers when both trees have the same

number of comparisons. The CPS algorithm can also be applied to the CFDT; however, the CFDT with classified points performs better than the CFDT with cluster centers.

References

1. J. R. Quinlan, "Induction of decision tree," *Mach. Learn.* **1**, 81–106 (1986).
2. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Vlassification and Regression Trees*, Wadsworth, Belmont, CA (1984).
3. A. Dobra and M. Schlosser, "Non-linear decision trees-NDT," in *Proc. 13th Int. Conf. Machine Learning (ICML'96)*, Bari, Italy, July 3–6 (1996).
4. S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artif. Intell. Res.* **2**, 1–32 (1994).
5. W. Pedrycz and Z. A. Sosnowski, "Designing decision trees with the use of fuzzy granulation," *IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans* **30**(2), 151–159 (2000).
6. R. Weber, "Fuzzy ID3: A class of methods for automatic knowledge acquisition," in *Proc. 2nd Int. Conf. Fuzzy Logic Neural Networks*, pp. 265–268 (1992).
7. S. B. Gelfand, C. S. Ravishankar, and E. J. Delp, "An iterative growing and pruning algorithm for classification tree design," *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(2), 163–174 (1991).
8. O. T. Yildiz and E. Alpaydin, "Omnivariate decision trees," *IEEE Trans. Neural Netw.* **12**(6), 1539–1546 (2001).
9. H. Zhao and S. Ram, "Constrained cascade generalization of decision trees," *IEEE Trans. Knowl. Data Eng.* **16**(6), 727–739 (2004).
10. M. M. Gonzalo and S. Alberto, "Using all data to generate decision tree ensembles," *IEEE Trans. Syst. Man Cybern., Part C Appl. Rev.* **34**(4), 393–397 (2004).
11. Y. Yuan and M. J. Shaw, "Introduction of fuzzy decision trees," *Fuzzy Sets Syst.* **69**, 125–139 (1995).
12. J. R. Quinlan, "Introduction of decision trees," *Mach. Learn.* **1**, 81–106 (1986).
13. H. Ichihashi, T. Shirai, K. Nagasaka, and T. Miyoshi, "Neuro-fuzzy ID3," *Fuzzy Sets Syst.* **81**, 157–167 (1996).
14. X. Wang, B. Chen, G. Qian, and F. Ye, "On the optimization of fuzzy decision trees," *Fuzzy Sets Syst.* **112**, 117–125 (2000).
15. X. Z. Wang, D. S. Yeung, and E. C. C. Tsang, "A comparative study on heuristic algorithms for generating fuzzy decision trees," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.* **31**(2), 215–226 (2001).
16. W. Pedrycz and Z. A. Sosnowski, "Designing decision trees with the use of fuzzy granulation," *IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans* **30**(2), 151–159 (2000).
17. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function*, Plenum, New York (1981).
18. A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.* **31**(3), 264–323 (1999).
19. W. Pedrycz and Z. A. Sosnowski, "C-fuzzy decision trees," *IEEE Trans. Syst. Man Cybern., Part C Appl. Rev.* **35**(4), 498–511 (2005).
20. C. Merz and P. Murphy, *UCI Repository of Machine Learning Databases*, Dept. of CIS, Univ. of California, Irvine, http://www.ics.uci.edu/~mlearn/MLRepository.html (2006).
21. P. Borodatz, *Textures—A Photographic Album for Artists and Designers*, Dover, New York (1966).
22. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Boston (1992).

Shiueng-Bien Yang received his BS in 1993 and PhD in 1999, both from the Department of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan. He is currently a professor of Department of Computer Science and Information Engineering, Leader University, Tainan City, Taiwan. Yang's research interests include pattern recognition, speech coding, image processing, and neural network.