Journal of Computational and Applied Mathematics 371 (2020) 112710

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Uncertain decision tree for bank marketing classification

Shiueng-Bien Yang *, Tai-Liang Chen

Department of Digital Content Application and Management, Wenzao Ursuline University of Languages, 900 Mintsu 1st Road Kaohsing 80424, 886, Taiwan

ARTICLE INFO

Article history:
Received 30 September 2019
Received in revised form 22 December 2019

Keywords:
Decision trees
Genetic clustering algorithm
Uncertain data
Merging algorithm

ABSTRACT

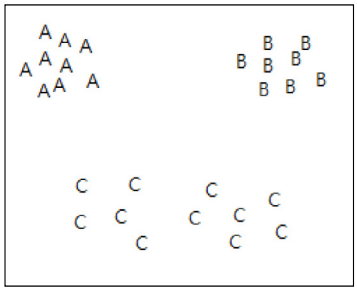
This study proposes a novel decision tree for uncertain data, called the uncertain decision tree (UDT), based on the uncertain genetic clustering algorithm (UGCA). UDT extends the decision tree to handle data with uncertain information, in which the uncertainty must be considered to obtain high quality results. In UDT, UGCA automatically searches for the proper number of branches of each node, based on the classification error rate and the classification time of UDT. Restated, UGCA reduces both the classification error rate and computing time and, then, optimizes the proposed UDT. Before the UDT is designed using UGCA, an uncertain merging algorithm (UMA) is also developed to reduce the uncertain data set, thereby allowing UGCA to process a large data set efficiently. Importantly, experimental results demonstrate that the proposed UDT outperforms traditional uncertain decision trees.

1. Introduction

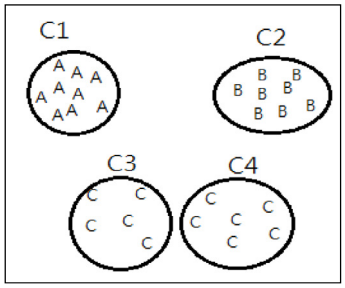
Decision trees [1–16] are the most widely used technique for the machine learning, pattern recognition and classification tasks. The traditional C4.5 decision tree can be deployed on a discrete class of problems [13]. Many decision trees are developed that are efficient to alleviate the shortcoming allied with C4.5 decision trees. Fuzzy C-mean (FCM) was well explained by Bezdek [14] and Pedrycz and Sosnowski [15]. The FCM is an iterative optimization process in which we iteratively update the partition matrix and prototypes until some termination criterion has been satisfied. However, the disadvantage of FCM is very sensitive to initialization. In 2005, the fuzzy C-means decision tree (CFDT) [16] was designed based on FCM. The CFDT is multi-feature decision tree that effectively classifies the samples with multi-feature entities characterized by high homogeneity (low variability). In 2013, the fuzzy clustering based decision tree (FCBDT) was proposed for large population speaker identification [17]. In Ref. [17], the fuzzy clustering algorithm liked the FCM was proposed to design the binary decision tree. In data mining, many genetic algorithms (GA) were proposed to build a decision tree for classification. However, when the optimal solution is a very full or narrow tree or the structure of it is very sparse, it is very hard for GA to get a satisfying result efficiently [18,19]. In 2011, the multiage genetic programming algorithm (MGP), was proposed to deal with the problems of searching decision tree with maximal classification accuracy [20]. However, the computing time of the decision tree is not considered during the design of MGP. In 2012, the fuzzy decision tree based on the genetic algorithm (GA), namely GCFDT, was proposed, and GCFDT outperforms traditional C4.5 and fuzzy decision tree [21]. In GCFDT, the center of each cluster is a real number, which is directly encoded in the bit string of GA. Thus, the number of clusters ($k = 2$) generated by GA is fixed, and the binary decision tree is designed. There are two drawbacks in the mentioned trees. First, the mentioned trees are fixed-branches

* Corresponding author.
E-mail addresses: 98010@mail.wzu.edu.tw (S.-B. Yang), 97007@mail.wzu.edu.tw (T.-L. Chen).

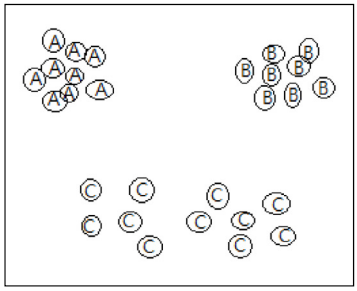
https://doi.org/10.1016/j.cam.2020.112710
0377-0427/



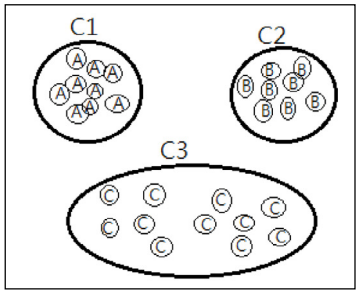
(a) The original uncertain data set.



(b) Four clusters are generated without considering the uncertainty.



(c) Each sample is various within a range.



(d) Three clusters are generated by our method.

Fig. 1. An example of clustering result for the uncertain data.

decision trees. That is, each node is split into a fixed number of branches during the design of decision trees. That is, the data set contained in a node is divided into a fixed number of clusters. However, cases usually occur where it is inappropriate to divide the set of training samples into a fixed number of clusters. Next, although the classification error rate can be reduced by splitting a node to grow the decision tree, the average number of comparisons is also increased when the tree is depth. However, the computing complexity is not considered during the design of the mentioned decision trees.

The uncertain data means lack of certainty. The data uncertainty comes by different parameters including sensor error, network latency measurements precision limitation and multiple repeated measurements. Numerous uncertain data classification algorithms have been proposed in the literature in recent years. There has been a growing interest in uncertain data mining. The well-known k-means clustering algorithm has been extended to the UK-means algorithm [22]. To improve the performance of UK-means, pruning techniques have been proposed [23,24]. In 2016, the traditional decision tree algorithm is modified by combining firefly and weighted entropy measure [25]. In [25], an efficient node splitting scheme was proposed for decision tree construction. In 2018, an induced model adapted to uncertainty data was proposed [26]. In [26], both a prediction and split rule for a tree construction taking into account the uncertainty of each quantitative observation from the data base. From the methods of the construction of a tree described above, they are all based on the minimization of a risk function and can be summarized into two tasks: a split rule and a prediction rule [27,28]. In [29], the density-based classification was proposed. Density-based classification requires that the joint probability distribution of the data attributes be known. Also, decision tree classification on uncertain data has been addressed for decades in the form of missing values [30,31]. Also, the fuzzy information models data uncertainty arising from human perception and understanding [31]. The uncertainty reflects the ambiguity of concepts. Fig. 1 presents an illustrative example of the uncertain data set. Fig. 1(a) shows three classes labeled by the symbols, 'A', 'B' and 'C'. In Fig. 1(b), the data set is divided into four clusters, C_1, C_2, C_3 and C_4, by the clustering algorithm without considering the uncertainty. However, class 'C' does not need to be classified into two clusters, C_3 and C_4, as shown in Fig. 1(b). Fig. 1(c) indicates that each sample varies within a spherical shape of range, when the uncertainty of each sample is considered. In Fig. 1(d), the proposed method divides the uncertain data set into three clusters, C_1, C_2 and C_3.

The novelties of this paper are described as follows. (1) This paper proposes a genetic clustering algorithm for uncertain data, called uncertain genetic clustering algorithm (UGCA). The UGCA is proposed to design the decision tree for uncertain data. In UGCA, the users need not determine the number of branches of the nodes in the decision tree. UGCA automatically

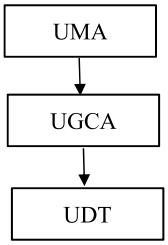


Fig. 2. The structure of the proposed research method.

searches for the proper number of branches of each node, based on the classification error rate and the computing time of UDT. Furthermore, this paper proposes the merging algorithm for uncertain data, called the uncertain merging algorithm (UMA). The main goal of UMA is to reduce the size of uncertain data set before the UGCA is applied to design the decision tree. Thus, UGCA can process the large data set efficiently. (2) This paper proposes the decision tree for uncertain data, called uncertain decision tree (UDT), based on the UGCA and UMA. In UDT, the number of branches of each node is determined by UGCA. Thus, the UDT is a variable-branches decision tree. Furthermore, the UDT is extended to handle uncertain data. Data uncertainty plays a major role in various applications. When UDT is applied to uncertain data, the latter must be considered to obtain high quality results. Fig. 2 shows the structure of the proposed research method. In Fig. 2, the UMA is used to reduce the size of the training data set, and then the UGCA can process the data set efficiently when the UDT is designed by the UGCA.

The remainder of this paper is organized as follows. The design of the UMA is presented in Section 2. Section 3 describes the design of UGCA. The design of UDT is described in Section 4. Subsequently, experiments are given in Section 5, and finally Section 6 presents conclusions.

2. Designs of UMA

Let $Q = \{(x_i, y_i) | i = 1, 2 \ldots, n\}$ denote the training data set that is used to design the UDT. In Q, $x_i \in R^m$ denotes the uncertain sample, and y_i denotes the corresponding class of uncertain sample x_i. Assume that Q is a large data set. The main goal of UMA is to reduce the size of the set Q. That is, the two closed uncertain samples that belong to the same class can be merged at a time to form a new cluster in UMA. Before the UMA is designed, the uncertain distance is described as follows.

An uncertain sample x_i is an uncertain sample associated with a probability density function, f, where $f(v)$ denotes the probability for x_i to assume a certain vector, v. Here, the Gaussian distribution is chosen for representing uncertainty. Let $BR_i(x_i)$ denote the set of values $\{w \in D | d(x_i, w) \leq R_i\}$, i.e. hyperball $BR_i(x_i)$ with center x_i and radius R_i. Namely, x_i refers to a variant within the hyperball with radius R_i. Restated, assume the existence of a cluster, $C(x_i)$, with radius R_i, which contains all of the variants of x_i. Therefore, each uncertain sample, x_i, has a corresponding cluster $C(x_i)$ containing x_i. Let $Class(C(x_i))$ denote the class of cluster $C(x_i)$. Then,

$$Class(C(x_i)) = y_i, \text{ for } i = 1, 2, \ldots, n. \tag{1}$$

The mean value (center) $\bar{x}_i$ of $C(x_i)$ is given as

$$\bar{x}_i = \int_{v \in BR_i(x_i)} v f(v) dv. \tag{2}$$

The users usually spend much time in calculating the integration in Eq. (2). In this work, when the training data set Q is sufficiently large, the set of neighbors of x_i, $N(x_i)$, is regarded as the set of variants of x_i. Where $N(x_i)$ is given as

$$N(x_i) = \{x_k \mid \|x_i - x_k\| \leq R_i, y_k = y_i\}, \tag{3}$$

where $\|\bullet\|$ denotes the Euclidean distance. Thus, Eq. (2) can be rewritten as

$$\bar{x}_i = \sum_{\substack{x_k \in N(x_i) \\ x_k \in Q}} x_k f(x_k). \tag{4}$$

Clearly, users take less time to compute $\bar{x}_i$ in Eq. (4) than in Eq. (2). Thus, the distance, $d(C(x_i), C(x_j))$, between two clusters, $C(x_i)$ and $C(x_j)$, is defined as

$$d(C(x_i), C(x_j)) = \left\| \bar{x}_i - \bar{x}_j \right\|. \tag{5}$$

A situation in which $d(C(x_i), C(x_j)) \leq R_i + R_j$ denotes that both clusters, $C(x_i)$ and $C(x_j)$, have an intersection. In this case, this situation denotes that these two clusters, $C(x_i)$ and $C(x_j)$, are sufficiently close and, then, they are considered to be merged while $Class(C(x_i)) = Class(C(x_j))$. Otherwise, these two clusters, $C(x_i)$ and $C(x_j)$, are separated.

The merging algorithm for uncertain data, UMA, is described as follows.

Algorithm: UMA

Input: The set, Q, consists of n training samples, each of which is represented by the pair (x_i, y_i), for $1 \le i \le n$. Where y_i denotes the corresponding class of uncertain sample x_i. The value of u.

Output: m clusters. ($m \ll n$)

Step 1. For $1 \le i \le n$, do the following.
Calculate the distance d_i between x_i and its nearest neighbor.

$$\gamma_i = \arg \min_{\substack{1 \le j \le n \\ i \ne j}} \left\| x_i - x_j \right\|, \tag{6}$$

$$d_i = \left\| x_i - x_{r_i} \right\|, \tag{7}$$

Step 2. Find the radius R_i for $1 \le i \le n$ as follows.

$$R_i = u d_i. \ (u \text{ is a weight.}) \tag{8}$$

Step 3. For $1 \le i \le n$, do the following.

Step 3.1 Generate the cluster, $C(x_i)$, with radius R_i, which contains x_i and samples in $N(x_i)$. Also, $N(x_i)$ is calculated as Eq. (3).

Step 3.2 Calculate the mean value (center), $\bar{x}_i$, of $C(x_i)$, according to Eq. (4).

Step 4. For $1 \le i \le n$, perform the following steps.

If $i \ne j$, for $1 \le j \le n$, compute $d(C(x_i), C(x_j))$ according to Eq. (5). Otherwise, set $d(C(x_i), C(x_j)) = \infty$.

Step 5. View these n training samples (clusters) as nodes of a graph. Set each element $A(i, j)$ in the adjacency matrix $A_{n \times n}$ as follows.

$$A(i,j) = \begin{cases} 1 & \text{if } d(C(x_i), C(x_j)) \le R_i + R_j \text{ and } y_i = y_j \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

where $1 \le i \le n$ and $1 \le j \le n$.

Step 6. Represent the training samples (clusters) by the nodes in the graph. Find the connected nodes of this graph. Each connected node forms a large cluster. Assume that m clusters, $B_1, B_2, \ldots, B_m$, are obtained. Let b_i denote the center of cluster B_i for $1 \le i \le m$. Each center, b_i, can then be calculated as

$$b_i = \sum_{x_k \in B_i} x_k f(x_k) \text{ for } 1 \le i \le m. \tag{10}$$

Then, m clusters, $B_1, B_2, \ldots, B_m$, serve as the output, according to the adjacency matrix, A.

In the UMA, the value of u is not critical. When the value of u is large, the UMA produces less clusters and each cluster is large, Conversely, if the value of u is small, the UMA generates more clusters and each cluster is small. After the UMA is finished, these m clusters, $B_1, B_2, \ldots, B_m$, are as the input of UGCA. Each cluster B_i, for $1 \le i \le m$, is regarded to be a component and will not be divided. These m clusters (components) will be merged in UGCA. That is, only m ($m \ll n$) components must be further processed in UGCA. Thus, UGCA can efficiently process the large data set.

3. Designs of UGCA

Let t be the leaf node of UDT, and let t contain m components, $B_1, B_2, \ldots, B_m$, obtained from the UMA. The following describes how to generate the child nodes of t by using the UGCA. Fig. 3 shows the flow chart of genetic algorithm. UGCA is designed based on the genetic algorithm that consists of an initialization step and iterations with three phases, reproduction, crossover and mutation, in each generation. The details are described in the following.

In the initialization step of UGCA, a population of N strings is randomly generated. The length of each string is m. N strings are generated such that the 1s in the strings are uniformly distributes within $[1, m]$. Each bit string represents a set $\{B_1, B_2, \ldots, B_m\}$. Let $I = (I_1, I_2, \ldots, I_m)$ be a bit string in the population. Let Q_1 be the set that contains the corresponding components, B_i ($I_i = 1$), and let Q_2 be the set that contains the corresponding components, B_j ($I_j = 0$). Each B_i in Q_1 is a seed to generate a cluster. Let the set, Q_1, of the string I generate n clusters, C_i for $1 \le i \le n$. That is, n child nodes of t are generated when node t is split.

The main design issue of reproduction phase is to design the fitness function for the string I. Before the fitness function of the string I is defined, we describe how to generate these n clusters, C_i for $1 \le i \le n$. Initially, each cluster C_i contains only one component, B_i, and then the center S_i of cluster C_i is set to the center of B_i. Let b_i denote the center of B_i. Then,

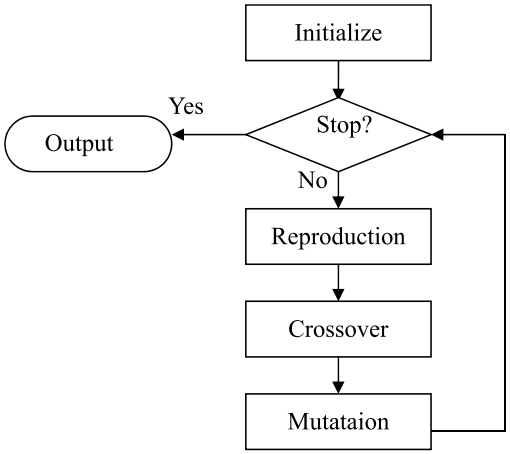


Fig. 3. The flow chart of the genetic algorithm.

we set $S_i = b_i$. The components in Q_2 are considered one at a time to search for the nearest cluster. The distance, $D(C_i, B_j)$, between the component, B_j, and the cluster, C_i, is defined as

$$D(C_i, B_j) = \sum_{\substack{x_k \in B_j \\ B_j \in Q_2}} \|x_k - S_i\| f(x_k). \tag{11}$$

where x_k denotes the kth uncertain sample contained in B_j. Thus,

$$B_j \subset C_{i^*} \text{ if } i^* = \arg\min_{1 \le i \le n} D(C_i, B_j). \tag{12}$$

If B_j is classified into the cluster C_{i^*} to form the new cluster $\hat{C}_{i^*}$, then the new center $\hat{S}_{i^*}$ of cluster $\hat{C}_{i^*}$ is updated as

$$\hat{S}_i = \sum_{B_k \subset \hat{C}_{i^*}} b_k f(b_k), \tag{13}$$

After the components in Q_2 have all been considered, n clusters, C_i for $1 \le i \le n$, are obtained from the string I. Then, the fitness function of string I is defined as,

$$Fitness(I) = \lambda(t, n), \tag{14}$$

where $\lambda(t, n)$ is defined as Eq. (22), which is described in the next section. In Eq. (14), the fitness function is designed based on both of the classification error rate and the computing time of UDT. The growing method of UDT emphasizes that the fitness function must be as large as possible for a split node t. After the fitness of each string in the population is calculated, the reproduction operator is implemented using a roulette wheel with slots sized according to fitness.

In the crossover phase, the crossover operator is applied to a selected pair of strings I and J, then two random numbers e and f in $[1, m]$ are generated to decide which pieces of the strings are to be interchanged. After the crossover phase, two new strings, I' and J', replace the strings, I and J, in the population. The meaning of crossover is the exchange of cluster seeds between different strings. Finding a suitable combination of cluster seeds can enhance the fitness of the string.

In the mutation phase, the bits of the strings in the population are chosen from $[1, m]$ with a probability. Each chosen bit is then changed from 0 to 1 or from 1 to 0. After the mutation phase, the new string I' can be obtained and replace the original string I. The meaning of mutation is to let the selected training sample have the opportunity to become (from 0 to 1) or cancel (from 1 to 0) the cluster seed, which helps the UGCA to find a best solution.

4. Designs of UDT

In UDT, the growing method selects a leaf node to split according to both of the classification error rate and the computing time. Before the growing method of UDT is described, the computing complexity and the classification error rate of UDT are defined as follows.

The computing complexity of UDT, T, is defined as follows. Let t' be an internal node in T, and let t' have $r(t')$ child nodes (branches). That is, the node t' contains $r(t')$ vectors that are used to classify the input sample. Then, the computing complexity of internal node, t', is $r(t')$. Let the set, H, consist of h leaf nodes, $t_1, t_2, \ldots, t_h$, in UDT. Let t_i be a leaf node,

and let $L(t_i)$ denote the set of internal nodes that are required to be calculated when the input sample travels the UDT from the root node to the leaf node t_i. Then, the computing complexity of the leaf node t_i, $V(t_i)$, is defined as

$$V(t_i) = \sum_{t'_j \in L(t_i)} r(t'_j). \tag{15}$$

Let $P(t_i)$ represent the probability on the training samples in the leaf node t_i. Thus, we have

$$\sum_{t_i \in H} P(t_i) = 1, \quad 0 \le P(t_i) \le 1. \tag{16}$$

Finally, the computing complexity of T, $V(T)$, is defined as

$$V(T) = \sum_{t_i \in H} P(t_i) V(t_i). \tag{17}$$

Next, the classification error rate of UDT, T, is defined as follows. Let $Class(B_k)$ denote the class of component B_k. Let the cluster, C_i, is a set of $\{(B_k, Class(B_k)) \,|B_k \in C_i\}$ that contained in the leaf node t_i. That is, the cluster C_i is not a pure class when the components contained in C_i have different class. The distance, $D(C_i, B_j)$, between the component B_j and the cluster C_i is defined as Eq. (11). Then, the probability, $P(C_i, B_j)$, B_j is classified to C_i, is given as

$$P(C_i, B_j) = \frac{1}{\sum_{k=1}^{h} \left[\frac{D(C_i, B_j)}{D(C_k, B_j)} \right]}, \text{ for } 1 \le i \le h. \tag{18}$$

Notably, $P(C_i, B_j)$ satisfies the constraints $0 \le P(C_i, B_j) \le 1$ and $\sum_{k=1}^{h} P(C_k, B_j) = 1$. Then, the representative of t_i, positioned in the output space is given as

$$M_i = \frac{\sum_{(B_j, Class(B_j)) \in C_i} P(C_i, B_j) Class(B_j)}{\sum_{(B_j, Class(B_j)) \in C_i} P(C_i, B_j)}. \tag{19}$$

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

$$R(t_i) = \sum_{(B_j, Class(B_j)) \in C_i} P(C_i, B_j)(Class(B_j) - M_i)^2 \tag{20}$$

Finally, the classification error rate of T, $R(T)$, is defined as

$$R(T) = \sum_{t_i \in H} P(t_i) R(t_i) \tag{21}$$

The growing method of UDT is described as follows. Fig. 4 shows an example of this process. In Fig. 4, T_1 denotes the decision tree after the leaf node t_1 is split in the tree, T, and T_2 denotes the decision tree after the leaf node t_2 is split in the tree, T. Fig. 4 shows that T_2 is better than T_1 because the classification error rate of T_2 is smaller than that of T_1 when they have the same computing time. In UDT, the growing method selects one node with the maximal value of $\frac{\Delta R}{\Delta V}$ to split in T, where ΔR and ΔV denote the variation in the classification error rate and average computing time, respectively. Let the new tree T' indicate the tree T after node t_i is split into n_i child nodes. Then,

$$\lambda(t_i, n_i) = \frac{\Delta R}{\Delta V} = \frac{R(T) - R(T')}{V(T') - V(T)}, \tag{22}$$

where $R(T)$ is defined as Eq. (21) and $V(T)$ is defined as Eq. (17). In UDT, the design of the growing method is to select the leaf node with the largest $\lambda(t_i, n_i)$ to be split at a time.

In the following, the design of UDT with the storage constraint is described.

Algorithm: Design_UDT

Input: The storage threshold δ and the training data set.
Output: The UDT with the storage constraint.
Step 1. Let the root node of tree T contain all of the training samples in the training data set. Set $STORAGE = 0$. Let H denote the set of leaf nodes in T. Initially, the set of H contains the root node.
Step 2. **While** $STORAGE < \delta$

Step 2.1. For each node t_i in H and $R(t_i) > 0$, perform the following.

Step 2.1.1 The UGCA is applied to the node t_i and generates n_i child nodes of t_i.
Step 2.1.2 Calculate the value of $\lambda(t_i, n_i)$.

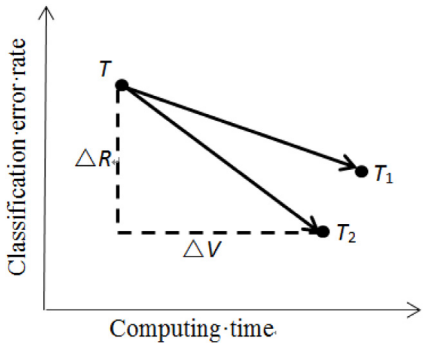


Fig. 4. An example illustrating that T_2 is better than T_1.

Table 1
Description of data sets.

Data set	Num. features	Num. vectors	Num. 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	10 992	10
Letter	16	20 000	26
Speech	480	6240	26
Bank marketing	20	41 188	2

Step 2.2. Let $k = \arg\max_{t_i \in H} \lambda(t_i, n_i)$, and let the node t_k be divided into n_k child nodes. The growing method selects the node t_k to split, and then the new decision tree T' is generated. Add these n_k nodes to the set of H.

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

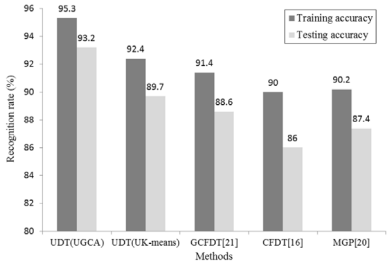
Step 3. Output the UDT with storage constraint, T.

5. Experiments

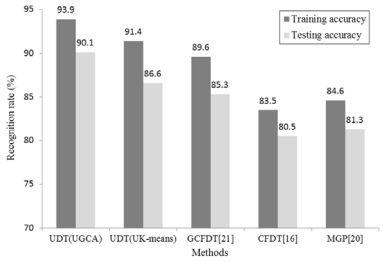
In the experiments, UDT based on the UGCA is compared with CFDT [16] based on the FCM. To test the performance of UGCA, UDT is also compared with the two decision trees, MGP [20] and GCFDT [21], that are based on the genetic algorithms. Furthermore, the proposed UGCA is compared with the UK-means algorithm for uncertain data. Ten data sets, comprising eight from the UC Irvine repository [32], speech data set extracted from the ISOLET database and bank marketing data set extracted from the UC Irvine repository [33], were adopted in this experiment. Table 1 lists the features of ten data sets. 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. In the bank marketing data set, the samples in the bank-additional-full.csv are used in the experiments. The bank-additional-full.csv contains 41 188 samples and each sample has 20 features. The samples in the bank-additional-full.csv are randomly picked half of the samples for training and the other half for testing.

In the experiments, the value of u is set to 2 in UMA. In UGCA, the population size is set to 200, and the crossover and mutation probabilities are set to 80% and 5%, respectively. The UGCA is run over 500 generations, and the best solution obtained from these 500 generations is retained. These methods are compared by designing the decision trees for each data set, based on the same number of nodes (storage). Table 2 lists the number of nodes used to design UDT and other methods.

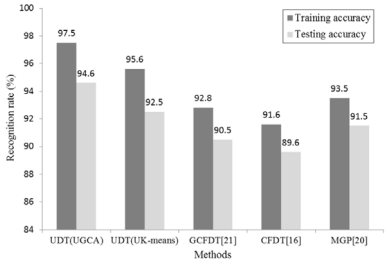
Fig. 5 shows the recognition rates of UDT and other methods. According to this figure, UDT(UGCA) and UDT(UK-means) indicate that the trees are designed by using the proposed UGCA and the UK-means algorithm, respectively. In the experiments, after UDT(UGCA) is designed, the UDT(UK-means) is then designed as the same structure of UDT(UGCA). Based on Fig. 5, we can infer the following. (1) UDT(UGCA) outperforms CFDT, MGP and GCFDT, indicating that variable-branch decision tree (eq. UDT(UGCA)), performs better than that of fixed-branch decision trees (e.g., CFDT, MGP and



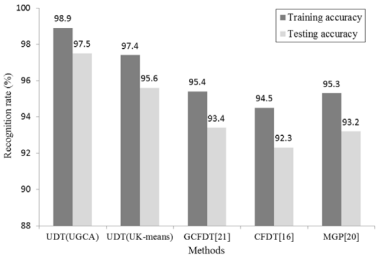
(a) Glass data set.



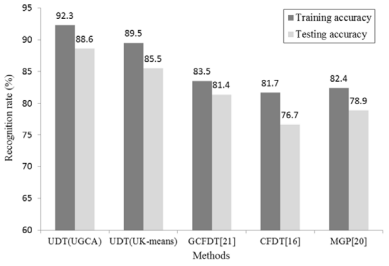
(b) Hepatitis data set



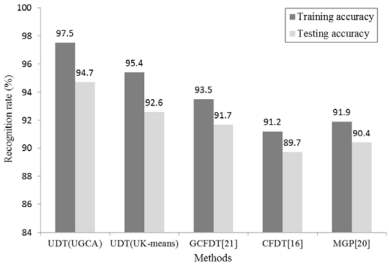
(c) Ionosphere data set.



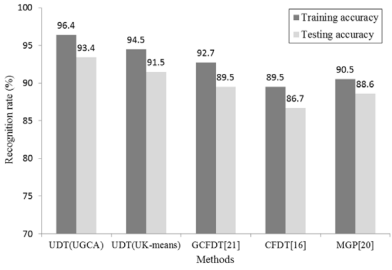
(d) Iris data set.



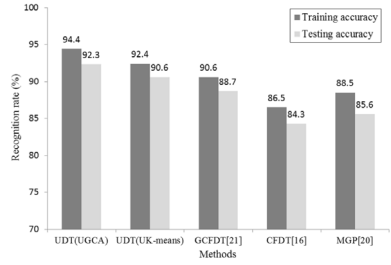
(e) Pima data set.



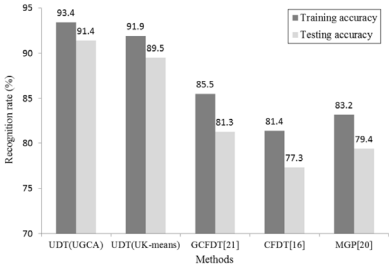
(f) Abalone data set.



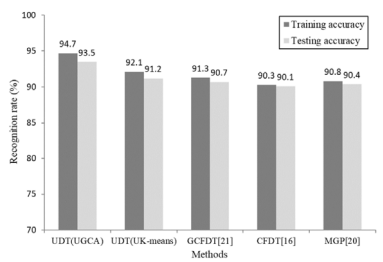
(g) Pendigits data set



(h) Letter data set



(i) Speech data set



(j) Bank marketing

Fig. 5. A comparative analysis for several methods.

Table 2
The number of nodes used to design the UDT.

Data set	Num. nodes
Glass	18
Hepatitis	14
Ionosphere	22
Iris	11
Pima	26
Abalone	54
Pendigits	48
Letter	52
Speech	62
Bank marketing	68

GCFDT). Our results suggest that each node need not split into the same number of branches (or binary trees) during designing the decision trees. Furthermore, UDT(UGCA) is designed for the uncertain data, whereas that the other methods are designed for the certain data. For example, in Fig. 5(j), although all methods have a recognition rate of more than 90%, the testing accuracy of UDT(UGCA), 93.5%, that is higher than that of other methods in the bank marketing data set. That is, the uncertainty must be considered to obtain high quality results in the bank marketing data set. (2) This study evaluates the performance of UGCA by comparing UDT(UGCA) with three methods, UDT(UK-means), MGP and GCFDT. According to Fig. 5, UDT(UGCA) outperforms three methods, UDT(UK-means), MGP and GCFDT. This denotes that the design of UGCA is based on the classification error rate and the computing time of UDT, whereas that of three methods, UDT(UK-means), MGP and GCFDT, does not. In the UDT(UGCA), the decision tree is grown by selecting the node with the largest value of λ defined as Eq. (22) to be split. Experimental results demonstrate that the UDT(UGCA) outperforms other methods.

6. Conclusions

This study proposes a novel UDT for uncertain data. The proposed UDT is a variable-branches decision tree, which is more efficient than that of traditional fixed-branch decision trees. UDT extends the decision tree to handle data with uncertain information. The high quality results can be obtained in UDT since the uncertainty is considered in UDT. Also, this study proposes the UGCA for uncertain data. The UGCA can automatically search for the appropriate number of branches of each splitting node in UDT according to the classification error rate and the computing time. Furthermore, the UMA is proposed for merging uncertain data. The UMA attempts to reduce the uncertain data set before applying UGCA to design the UDT. Thus, the UGCA can efficiently process the large data set. In our experiments, the UDT based on UGCA and UMA outperforms other methods.

References

[1] J.R. Quinlann, Induction of decision tree, Mach. Learn. 1 (1986) 81–106.
[2] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Vlassification and Regression Trees, Wadsworth, Belmont, CA, 1984.
[3] A. Dobra, M. Schlosser, Non-linear decision trees-NDT, in: Proc. 13th Int. Conf. Machine Learning (ICML'96), 1996, pp. 3–6.
[4] S.K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, J. Artif. Intell. Res. 2 (1994) 1–32.
[5] R. Weber, Fuzzy ID3; A class of methods for automatic knowledge acquisition, in: Proc. 2nd Int. Conf. Fuzzy Logic Neural Networks, 1992, pp. 265–268.
[6] S.B. Gelfand, C.S. Ravishankar, E.J. Delp, An iterative growing and pruning algorithm for classification tree design, IEEE Trans. Pattern Anal. Mach. Intell. 13 (3) (1991) 163–174.
[7] O.T. Yildiz, E. Alpaydin, Omnivariate decision trees, IEEE Trans. Neural Netw. 12 (6) (2001) 1539–1546.
[8] H. Zhao, S. Ram, Constrained cascade generalization of decision trees, IEEE Trans. Knowl. Data Eng. 16 (6) (2004) 727–739.
[9] M.M. Gonzalo, S. Alberto, Using all data to generate decision tree ensembles, IEEE Trans. Syst. Man Cybern. C 34 (4) (2004) 393–397.
[10] X. Wang, B. Chen, G. Qian, F. Ye, On the optimization of fuzzy decision trees, Fuzzy Sets and Systems 11 (2) (2000) 117–125.
[11] J.R. Quinlan, Introduction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[12] X.Z. Wang, D.S. Yeung, E.C.C. Tsang, A comparative study on heuristic algorithms for generating fuzzy decision trees, IEEE Trans. Syst. Man Cybern. B 31 (2) (2001) 215–226.
[13] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[14] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Functions, PlenUDMA, New York, 1981.
[15] W. Pedrycz, A. Sosnowski, Designing decision trees with the use of fuzzy granulation, IEEE Trans. Syst. Man Cybern. A 30 (2000) 151–159.
[16] P. Witold, A.S. Zenon, C-fuzzy decision trees, IEEE Trans. Syst. Man Cybern. C 35 (2005) 498–511.
[17] Y. Hu, D. Wu, A.o. Nucci, Fuzzy-clustering-based decision tree approach for large population speaker identification, IEEE Trans. Audio Speech Lang. Process. 21 (2013) 762–774.
[18] J. Daida, J. Polito, What makes a problem GP-hard? Analysis of a tunably difficult problem in genetic programming, Genet. Program. Evol. Mach. 2 (2) (2001) 165–191.
[19] J. Daida, H. Li, R. Tang, A. Hilss, What makes a problem GP-hard? Validating a hypothesis of structural causes, in: Proc. Genetic Algorithms Evol. Comput. Conf., in: Lecture Notes Computer Science, 2003, pp. 1665–1677.
[20] L. Yi, K. Wanli, A new genetic programming algorithm for building decision tree, Procedia Eng. 15 (3) (2011) 3658–3662.
[21] S.K. Shukla, M.K. Tiwari, GA guided cluster based fuzzy decision tree for reactive ion etching modeling: A data mining approach, IEEE Trans. Semicond. Manuf. 25 (1) (2012) 45–56.

[22] M. Chau, R. Cheng, B. Kao, J. Ng, Uncertain data mining: An example in clustering location data, in: Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD), 2006, pp. 199–204.
[23] W.K. Ngai, B. Kao, C.K. Chui, R. Cheng, M. Chau, K.Y. Yip, Efficient clustering of uncertain data, in: Proc. Int'l Conf. Data Mining (ICDM), 2006, pp. 436–445.
[24] S.D. Lee, B. Kao, R. Cheng, Reducing UK-Means to KMeans, in: Proc. First Workshop Data Mining of Uncertain Data (DUNE), in conjunction with the Seventh IEEE Int'l Conf. Data Mining (ICDM), 2007.
[25] K. Soundararajan, S. Sureshkumar, Decision tree approach for classifying uncertain data, World Eng. Appl. Sci. J. 7 (2) (2016) 74–84.
[26] Myriam Tami, M. Clausel, E. Devijver, E. Gaussier, J.-M. Aubert, et al., Decision tree for uncertainty measures. 2018. JDS 2018 : 50èmes Journées de statistique, Paris-Saclay.
[27] R. Genuer, J.-M. Poggi, Arbres CART et Forêts aléatoires, importance et sélection de variables, 2016, arXiv preprint 1610.08203.
[28] G. Louppe, Understanding Random Forests: From Theory to Practice, Thèse soutenue à l'université de Liège, 2014.
[29] H.-P. Kriegel, M. Pfeifle, Density-based clustering of uncertain data, in: Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2005, pp. 672–677.
[30] C.C. Aggarwal, On density based transforms for uncertain data mining, in: Proc. Int'l Conf. Data Eng. (ICDE), 2007, pp. 866–875.
[31] J.R. Quinlan, Learning logical definitions from relations, Mach. Learn. 5 (2) (1990) 239–266.
[32] C. Merz, P. Murphy, UCI Repository of Machine Learning Databases, Dept. of CIS, Univ. of California, Irvine, 2006.
[33] S. Moro, P. Cortez, P. Rita, A data-driven approach to predict the success of bank telemarketing, Decis. Support Syst. 62 (2014) 22–31, Elsevier.