Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5578682, 17 pages
https://doi.org/10.1155/2021/5578682

Research Article

Random Fuzzy Granular Decision Tree

Wei Li,[1] **Xiaoyu Ma**,[1] **Yumin Chen**,[1] **Bin Dai**,[1] **Runjing Chen**,[1] **Chao Tang**,[2] **Youmeng Luo**,[1] **and Kaiqiang Zhang**[1]

[1]*School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China*
[2]*School of Artificial Intelligence and Big Data, Hefei University, Hefei 230601, China*

Correspondence should be addressed to Wei Li; drweili@hotmail.com

Received 25 February 2021; Accepted 24 May 2021; Published 9 June 2021

Academic Editor: Venkatesan Rajinikanth

In this study, the classification problem is solved from the view of granular computing. That is, the classification problem is equivalently transformed into the fuzzy granular space to solve. Most classification algorithms are only adopted to handle numerical data; random fuzzy granular decision tree (RFGDT) can handle not only numerical data but also nonnumerical data like information granules. Measures can be taken in four ways as follows. First, an adaptive global random clustering (AGRC) algorithm is proposed, which can adaptively find the optimal cluster centers and maximize the ratio of interclass standard deviation to intraclass standard deviation, and avoid falling into local optimal solution; second, on the basis of AGRC, a parallel model is designed for fuzzy granulation of data to construct granular space, which can greatly enhance the efficiency compared with serial granulation of data; third, in the fuzzy granular space, we design RFGDT to classify the fuzzy granules, which can select important features as tree nodes based on information gain ratio and avoid the problem of overfitting based on the pruning algorithm proposed. Finally, we employ the dataset from UC Irvine Machine Learning Repository for verification. Theory and experimental results prove that RFGDT has high efficiency and accuracy and is robust in solving classification problems.

1. Introduction

The classification problem is a necessary research topic in data mining fields. Among classification approaches, the decision tree is very effective. It has the advantages of high classification accuracy, few parameters, and strong interpretability. Decision trees have been widely adopted in business, medical care, etc., and have achieved remarkable results. Data generated in daily life is increasing rapidly, which brings opportunities and challenges for decision trees to solve large-scale data classification problems. Decision tree is an inductive learning algorithm on the basis of examples. With the in-depth research on decision tree algorithms and the diversified needs in practical applications, a variety of learning algorithms or models for constructing decision trees have been proposed.

1.1. Information Entropy Decision Tree. Quinlan proposed the ID3 decision tree algorithm. This algorithm employs information gain in information theory as an evaluation of feature quality in the process of splitting tree nodes, and the feature with the largest information gain and the corresponding split point will be used for the construction of the node [1]. The ID3 algorithm is a clear and quick method, but it also has some obvious shortcomings. First of all, in the scope of processing datasets, it is not suitable for datasets containing continuous features; secondly, it tends to choose conditional features with more values as the optimal splitting features. In the same year, Schlimer and Fisher et al. proposed the ID4 decision tree method, which constructs decision trees in an incremental manner [2]. Two years later, ID5 algorithm was presented by Utgof et al., which allows the structure of the existing decision tree to be modified by adding new training instances without the need for retraining [3]. In addition, Xiaohu Liu and his colleagues also discussed the decision tree construction by considering both the information gain brought about by the conditional feature of the current node and the information gain of the conditional feature of the next node when selecting the split

feature [4]. Aiming at the shortcomings of ID3 decision trees in selecting features, Xizhao Wang et al. selected appropriate decision tree branches for merging during the construction of decision trees. This algorithm can increase the comprehensibility, improve the generality, and reduce the complexity [5]. The C4.5 decision tree improved performance compared with ID3 algorithm, proposed by Quinlan et al. in 1993 [6]. To solve the bias problem of information gain when selecting split features, this method employs the information gain rate as a metric for choosing split features. Also, the C4.5 algorithm also has the advantage of processing continuous features, discrete features, and incomplete datasets, which has stronger applicability than the ID3 algorithm. When constructing the C4.5 decision tree, the pruning operation is adopted, which can enhance the efficiency of the decision tree, reduce the scale of the decision tree, and effectively avoid the problem of overfitting. Domingos and his colleagues presented a very fast decision tree for data flow, which is called VFDT [7]. The algorithm shortens the training time of the decision tree by using sampling technology. Hulten designed CVFDT algorithm, which extended VFDT and required users to give parameters in advance [8]. Recently, some researchers have also proposed many other decision tree algorithms based on information entropy [9–16].

1.2. Gini Coefficient Decision Tree. In 1984, Breiman and his colleagues designed the CART algorithm, which adopts Gini coefficient as a metric for feature splitting, and chooses the conditional feature with the smallest Gini coefficient as the splitting feature of the tree node to generate a decision tree [17]. When generating a decision tree, if the purity of the spanning tree node is greater than or equal to the threshold assigned in advance, the tree node is stopped from dividing and then the main label of the instance data covered by the node is used as the label of the leaf node label. Meanwhile, the approach adopts resampling to analyze the accuracy of the constructed decision tree and perform pruning operations, and the decision tree with high accuracy and the smallest size is selected as the final decision tree. Here, the pruning method adopts the minimum cost complexity method, MCCP, which can solve the problem of overfitting, reduce the size, and improve interpretability. CART also has its own shortcomings. Due to the limitation of computer memory, this method cannot effectively process large-scale datasets. Aiming at the shortcomings of the CART algorithm in computer memory, in 1996 Mehta et al. proposed an SLIQ decision tree [18]. When building the decision tree, instances are presorted, and then the breadth-first method is used to choose the optimal split feature. The SLIQ algorithm has the advantages of fast calculation speed and ability to process larger datasets. When the data exceed the memory capacity, this method employs feature lists, classification lists, and class histograms to get the solution. However, when the amount of data is large to a certain extent, the algorithm still faces the problem of insufficient memory [19]. In 1998, Rastogi et al. presented a public decision tree algorithm [20]. The algorithm integrates the tree construction process with the tree adjustment process. By calculating the cost function value of the tree node, it is judged whether the node needs to be pruned or expanded. If it is not expanded, the node is marked as a leaf node. The combination of the establishment process and the adjustment process greatly increases the training efficiency of the public decision tree. The Rain Forest is a framework of the decision tree proposed by Gehrke and his colleagues in 2000 [21]. The research aim of the algorithm was to enhance scalability of decision tree and reduce the consumption of computer memory resources as much as possible.

1.3. Rough Set Decision Tree. Incorporating the rough set theory into the decision tree can make the decision tree have the ability to handle uncertain, incomplete, and inconsistent data. When using rough set to generate a decision tree, the main research focus is how to use rough set theory to choose node splitting features. Miao and his colleagues designed a rough set based on multivariate decision tree algorithm. The method first selects the conditional features in the kernel to construct a multivariate test, and then generates a new feature to split the node [22]. The advantage of this algorithm is that the training efficiency is relatively high, but because there are too many variables in the nodes of the decision tree, the interpretability of the decision tree is difficult. Wang et al. designed a fuzzy decision tree on the basis of rough set, which uses fuzzy integration to keep the results consistent [23]. In 2011, Zhai and his colleagues adopted the fuzzy rough set to generate a decision tree and presented a new selection criterion for fuzzy condition features [24]. Wei et al. constructed a decision tree according to a variable precision rough set, which allows small error in the classification process and improves the generalization ability of the decision tree [25]. Jiang and his colleagues designed an incremental decision tree learning method via rough set, and used this method to discuss the problem of network intrusion detection [26]. In 2012, Hu and others proposed a monotonous ordered mutual information decision tree. This decision tree employs the dominant rough set theory to establish a new metric to measure feature quality to construct tree nodes. This decision tree can be resistant to noise, handle monotonous classification problems, and has good effects on general classification problems [27]. On the basis of the algorithm, Qian and his colleagues designed a fusion monotonic decision tree. The algorithm uses feature selection technology to generate multiple data feature distributions to construct multiple decision trees, and employs these decision trees to make comprehensive decisions [28]. Pei and his colleagues designed a monotonously constrained multivariate decision tree. The algorithm first uses the ordered mutual information splitting criterion to generate different data subsets, and then optimizes these subdata to build a decision tree [29].

1.4. Parallel Decision Tree. Nonparallel or serial decision trees have received extensive and in-depth research and development, and a lot of decision tree models and algorithms have been proposed, but due to the recursive characteristics of decision trees and computing platforms,

relatively speaking, the research of decision trees parallelization is not very extensive. The following is a brief review and summary of the current research status of parallel decision trees. The research on parallel decision tree started with SPRINT proposed by Shafer et al. in 1996 [30]. The algorithm tries to avoid the problem of insufficient memory by improving data structure during the growth of decision tree, but during calculation of the tree splitting node, the algorithm requires a broadcast from the entire instance to the entire instance. Kufrin et al. discussed the parallelization of decision trees and introduced a parallelization framework in 1997 [31]. One year later, Joshi and his team proposed a parallel form of decision tree similar to SPRIENT. It is different from the traditional depth-first construction method of decision tree. The algorithm adopts a breadth-first form of decision tree growth within a parallel framework. This method can avoid possible load imbalance problem [32]. Srivastava and his team presented two parallel decision tree models on the basis of the synchronous construction approach and the asynchronous construction method, but both of these models have large communication overhead and load imbalance problems [33]. Shent et al. gave a parallel decision tree that divides the input instance into four subsets to build a decision tree, and applied the generated parallel decision tree to the user authentication problem [34]. With the in-depth research on parallel decision trees and the emergence of MapReduce distributed computing frameworks, Panda et al. designed a parallel decision tree in 2009, which relies on multiple distributed computing frameworks [35]. Walkowiak and his colleagues focused on the parallelization of decision trees, and proposed an optimization model for network computing for the distributed implementation of decision trees [36]. In 2012, Yin et al. adopted the scalable regression tree algorithm to give a parallel implementation of the Planet algorithm under the MapReduce framework [37]. In response to the problem of overlearning or overfitting, Wang Ran et al. proposed an extreme learning machine decision tree on the basis of a parallel computing framework, but the disadvantage of this algorithm is that it can only handle numerical datasets and cannot handle mixed dataset [38]. The parallel C4.5 decision tree proposed in [39] considers the problem of overfitting and mixed data types. Aiming at the ordered mutual information decision tree that is widely used in monotonic classification problems, Mu and his colleagues presented a fast version and gave its parallel implementation [40]. There are other parallel decision tree approaches proposed like distributed fuzzy decision tree [41], parallel Pearson correlation coefficient decision tree [42], etc. In addition to the above research, Li and other researchers proposed some classification and alignment algorithms [43–49] from the perspective of granular computing, which have good performance.

2. Contributions

In this study, a decision tree is constructed in granular space to solve the classification problem. The main contributions are as follows:

(i) We propose AGRC that can adaptively give the optimal cluster centers, which is a global optimization method and can avoid falling into local optimization solution.

(ii) We design the parallel granulation method based on the above clustering algorithm, which solves the problem of high complexity of traditional serial granulation and enhances the granulation efficiency.

(iii) In granular space, we define fuzzy granules and related operators and select features based on the information gain ratio to construct a fuzzy granular decision tree for classification. To avoid overlearning, we also design the corresponding pruning algorithm. The method presented can solve binary classification or multiclassification problem and give feature importance according to the order of the tree node generated.

3. The Problem

Let $S = (X, R, Y, f, V)$ be a classification system. The goal is to design a classification algorithm. During the process, parameters can be obtained by statistic instances. Then, the label of instance can be predicted by the model. The symbols mentioned above can be explained in detail as follows. $X = \{x_1, x_2, \ldots, x_n\}$ is an instance set. $R = \{r_1, r_2, \ldots, r_m\}$ represents a feature set. $V = \cup_{r \in R} V_r, V_r$ denotes the value region of feature r. h: $X \times R \longrightarrow V$ expresses an information function that allocates a value for each feature, that is, $\forall r \in R, x \in X, h(x, r) \in V_r$. $Y = \{y_1, y_2, \ldots, y_l\}$ indicates a label set, where y_j is a label w.r.t. instance x_i.

4. The Algorithm Description

To obtain the solution of classification problem described in Section 3, the model can be presented as follows. First of all, to enhance the efficiency as much as possible, we need to cluster the data. During the process, an adaptive clustering algorithm is designed, which can obtain the quantity of cluster and the cluster centers automatically. Second, on the basis of the quantity of center and the cluster centers, a parallel granulation is executed by calculating the distance between instances and cluster centers and dividing instances set into instances subsets. Third, the problem of instance classification can be converted into the problem of granule classification in granular space. Fuzzy granules, related operators, and cost function are defined in granular space. Splitting nodes can be found by solving cost function to build a fuzzy granular decision tree. Fourth, the pruning algorithm w.r.t. RFGDT is designed. Finally, the label of test instance can be predicted by RFGDT. The overview is described in Figure 1.

4.1. Theory of AGRC. K-means algorithm is an unsupervised classification approach. The cluster centers and their numbers need to be specified in advance. The results obtained rely on the above cluster center. If the initial cluster

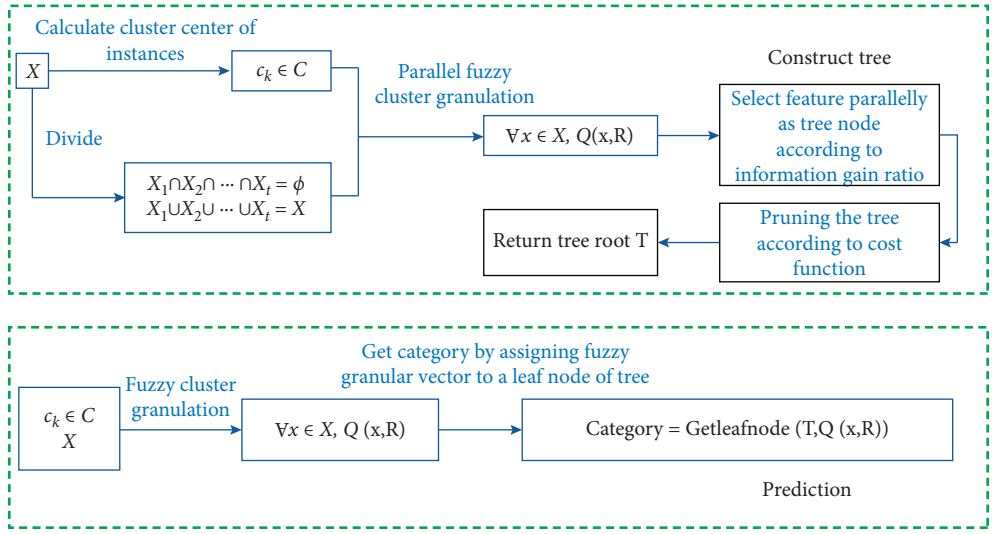

Figure 1: Overview of RFGDT.

center is not well selected, it will fall into a local optimal solution. An Adaptive Global Random Clustering algorithm is proposed, which adaptively selects cluster centers and their numbers, and the initial selection of cluster centers is random, which is a global optimization approach. The thought is as follows. We know that if the between-cluster variance α is large and the intracluster variance θ_k is small, then the performance is great. Hence, $\alpha^2/\sum_{k=1}^{K}\theta_k^2$ can be used as an evaluation indicator, where α represents standard deviation of intercluster and θ_k denotes standard deviation of $k-$th inner-cluster. The goal is to increase the ratio $\alpha^2/\sum_{k=1}^{K}\theta_k^2$ continuously until the maximum iteration is achieved. In each iteration, we have a set of new cluster centers and these parameters, including the quantity of cluster centers, cluster centers and evaluation values, are combined into an evaluation set. When the process is over, we can get cluster centers that are corresponding to the largest evaluation value in the evaluation set, which are the optimal parameters. Here, in each iteration, we select an instance as the next cluster center with a certain probability, until the quantity of cluster centers meets the preset in this iteration. The process is described as follows:

Step I: remove instances of missing feature values.

Step II: normalize feature values of instances into $[0, 1]$.

Step III: assign maximum iterations Max, evaluation set $E = \phi$ (composed of standard deviation of intercluster, standard deviation of inner-cluster, and evaluation value), and iteration $t = 1$.

Step IV: assign current cluster center set $C_t = \phi$ and the quantity of current cluster center $k = 0$ and generate a random number n within $(1, N)$ as the quantity of cluster center.

Step V: randomly select one instance x_i as a cluster center and set $k = k + 1$, $c_k = x_i$, $C_t = C_t \cup \{c_k\}$

Step VI: calculate the shortest distance between the remaining instances and all cluster centers

$$D(x_j) = \min\{\text{dist}(x_j, c_1), \ldots, \text{dist}(x_j, c_k)\}, \quad (1)$$

and the probability of an instance being selected as the next cluster center is

$$p(x_j) = \frac{D(x_j)^2}{\sum_{i=1}^{n} D(x_i)^2}. \quad (2)$$

Step VII: if x_j is selected as cluster center, we set $k = k + 1$, $c_k = x_j$, $C_t = C_t \cup \{c_k\}$.

Step VIII: if $k < n$, go to Step V; otherwise, go to Step IX.

Step IX: calculate the standard deviation of intercluster and the standard deviation of inner-cluster and update the evaluation set, that is,

$$E = E \cup \{(\alpha_t, \theta_1, \ldots, \theta_k)\}. \quad (3)$$

Step X: update iteration $t \leftarrow t + 1$.

Step XI: if Max $< t$, go to Step XII; or else, go to Step IV.

Step XII: in evaluation set E, the cluster center set with the largest ratio of the intercluster standard deviation to the sum of inner-cluster standard deviation,

$$C^* = \{c_1, c_2, \ldots, c_{|C^*|}\} = \arg\max_{1 \le t \le \text{Max}} \frac{\alpha_t}{\sum_{j=1}^{|C_t|} \theta j}, \quad (4)$$

and the quantity of cluster center corresponding to it, $K = |C^*|$ (here, $|\cdot|$ expresses the quantity of elements of the set), is the optimal solution.

Step XIII: end.

The principle of Adaptive Global Random Clustering is given. On the basis of the principle, the algorithm is as shown in Algorithm 1.

4.2. From Data to Fuzzy Granules. Now, we introduce how to implement parallel fuzzy granulation of data via cluster

Input: instance set X, maximum iterations Max
Output: optimal cluster center set C^* and its number K
(1) Remove instances of missing feature values.
(2) Normalize feature values of instances into $[0, 1]$.
(3) $E = \phi$\\Initialize evaluation set.
(4) $t = 1$\\Set current iteration.
(5) While $t \le$ Max
(6) $C = \phi$\\ Let the current cluster center be an empty set.
(7) $n =$ Select Cluster Number $(1, N)$\\ Randomly generate the quantity of cluster center in $(1, N)$
(8) $k = 1$\\ Initialize the quantity of cluster center.
(9) $c_k =$ Random Select Instance $(X, 1)$\\ Randomly select an instance as a cluster center in the dataset.
(10) $C = C \cup \{c_k\}$
(11) While $k < n$
(12) $\forall j \in |X|, D(x_j) = \min\{\text{dist}(x_j, c_1), \ldots, \text{dist}(x_j, c_k)\}$
(13) $p(x_j) = D(x_j)^2 / \sum_{i=1}^{n} D(x_j)^2$. \\ \\Calculate the probability selected of x_j as the next cluster center.
(14) $p =$ GenProbability (); \\Randomly generate a probability.
(15) If $p(x_j) > p$ then $\{c_k = x_j; C = C \cup x_j; k = k + 1\}$
(16) End while \\ $k < n$
(17) $E = E \cup \{(\alpha_t, \theta_1, \ldots, \theta_k)\}$\\ The standard deviation of intercluster and one of inner-cluster are calculated and stored and updated in the evaluation set.
(18) $t = t + 1$.\\ Update current iterations.
(19) End while \\ $t <$ Max
(20) $C^* = \{c_1, c_2, \ldots, c_{|C^*|}\} = \text{argmax}_{1 \le t \le \text{Max}} \alpha_t / \sum_{j=1}^{|C_t|} \theta j$\\ In the evaluation set E, choose the cluster center set with the largest ratio.
(21) $C^* = \{c_1, c_2, \ldots, c_{|C^*|}\}$, $K = |C^*|$ (here, $|\cdot|$ represents the quantity of elements of the set). \\ Return optimal parameters.

Algorithm 1: Adaptive global random clustering.

centers. We can adopt AGRC to get the cluster center set $C = \{c_1, c_2, \ldots, c_k\}$. Let instance set be X and feature set be R. If $\forall x_i \in X, c_j \in C$, and $r \in R$, the similarity between c_j and x w.r.t. r is

$$d(x_i, c_j, r) = 1 - |h(x_i, r) - h(c_j, r)|, \tag{5}$$

where $0 \le h(x_i, r) \le 1$, $0 \le h(c_j, r) \le 1$, and $d(x_i, c_j, r)$ represents the similarity between x_i and c_j on r $(0 \le d(x_i, c_j, r) \le 1)$. A fuzzy granule generated by x_i can be written as

$$q(x_i, r) = \frac{d(x_i, c_1, r)}{c_1} + \frac{d(x_i, c_2, r)}{c_2} + \cdots + \frac{d(x_i, c_k, r)}{c_k}. \tag{6}$$

For simplicity, it can also be written as

$$q(x_i, r) = \sum_{j=1}^{k} \frac{d(x_i, c_j, r)}{c_j}. \tag{7}$$

Here "–" represents separator and "+" denotes the union of elements. In other words, $q(x_i, r)$ is the similarity between x_i and cluster centers. The cardinal number of fuzzy granule $q(x_i, r)$ is obtained by equation:

$$|q(x_i, r)| = \sum_{j=1}^{k} d(x_i, c_j, r). \tag{8}$$

Now we give operators of fuzzy granules. For $\forall x, z \in X$, the operators between fuzzy granules generated by z and x can be written as follows:

$$q(x, r) \vee q(z, r) = \sum_{j=1}^{k} \frac{a + b - ab - \min\{(1 - \lambda), a, b\}/\max\{\lambda, 1 - a, 1 - b\}}{c_j}, \tag{9}$$

$$q(x, r) \wedge q(z, r) = \sum_{j=1}^{k} \frac{ab/\max\{\lambda, a, b\}}{c_j}, \tag{10}$$

$$q(x, r) - q(z, r) = \sum_{j=1}^{k} \frac{a - b}{c_j}, \tag{11}$$

$$q(x, r) \oplus q(z, r) = q(x, r) \vee q(z, r) - q(x, r) \wedge q(z, r). \tag{12}$$

Here $a = d(x, c_j, r)$, $b = d(z, c_j, r)$, and λ is a parameter $(0 \le \lambda \le 1)$. For $\forall U \subseteq R$, $U = \{r_1, r_2, \ldots, r_{|U|}\}$, and $|U| \le |R|$, the fuzzy granular array generated by x on U can be written as follows:

$$. \; Q(x, U) = \frac{Q(x, r_1)}{r_1} + \frac{Q(x, r_2)}{r_2} + \cdots + \frac{Q(x, r_{|U|})}{r_{|U|}} = \sum_{i=1}^{|U|} \frac{q(x, r_i)}{r_i}. \tag{13}$$

The symbol "+" denotes union and the symbol "–" represents separator. The cardinal number of the fuzzy granular array can be written as

$$|Q(x, U)| = \sum_{i=1}^{|U|} |q(x, r_i)|. \tag{14}$$

Now, we have the operators between fuzzy granular arrays. Let $Q(x, U)$ and $Q(z, U)$ be the fuzzy granular arrays

generated by the instance x and z on feature subset U, respectively. The operators can be written as follows:

$$Q(x,U)\vee Q(z,U)=\sum_{i=1}^{|U|}\frac{q(x,r_i)\vee q(z,r_i)}{r_i}, \tag{15}$$

$$Q(x,U)\wedge Q(z,U)=\sum_{i=1}^{|U|}\frac{q(x,r_i)\wedge q(z,r_i)}{r_i}, \tag{16}$$

$$Q(x,U)-Q(z,U)=\sum_{i=1}^{|U|}\frac{q(x,r_i)-q(z,r_i)}{r_i}, \tag{17}$$

$$Q(x,U)\oplus Q(z,U)=\sum_{i=1}^{|U|}\frac{q(x,r_i)\oplus q(z,r_i)}{r_i}. \tag{18}$$

The difference between two fuzzy granular vectors can be written as follows:

$$d(Q(x,U),Q(z,U))=\frac{1}{|U|^*|C|}\sum_{r\in U}\frac{|Q(x,r)\oplus Q(z,r)|}{|Q(x,r)\vee Q(z,r)|}. \tag{19}$$

From information granulation, we can see that fuzzy granules and fuzzy granular arrays are generated by their operators. The fuzzy granules consist of the space called fuzzy granular space.

Theorem 1. *For $\forall x_i, x_j \in X$, $\forall U \subseteq R$, $\forall r \in U$, the similarity of fuzzy granules satisfies the following equation:*

$$0\le d\left(Q(x_i,U),Q\left(x_j,U\right)\right)\le 1. \tag{20}$$

Proof 1. According to equation (7), we have $q(x_i,r)=\sum_{t=1}^{k}d(x_i,c_t,r)/c_t$ and $q(x_j,r)=\sum_{t=1}^{k}d(x_j,c_t,r)/c_t$. According to equation (5), we have $\forall x\in X, 0\le d(x,c_t,r)\le 1$. As a sequence, $0\le d(x_i,c_t,r)\le 1$ and $0\le d(x_j,c_t,r)\le 1$ are established. Due to $|q(x_i,r)|=\sum_{c_t\in C}d(x_i,c_t,r)$ and $|q(x_j,r)|=\sum_{c_t\in C}d(x_j,c_t,r)$, we can have $0\le|q(x_i,r)|\le|C|$ and $0\le|q(x_j,r)|\le|C|$. Equations (13) and (14) show $Q(x_i,U)=\sum_{t=1}^{|U|}q(x_i,r_t)/r_t$ and $|Q(x_j,U)|=\sum_{t=1}^{|U|}|q(x_j,r_t)|$. Therefore, $0\le|Q(x_i,U)|\le|U|^*|C|$ and $0\le|Q(x_j,U)|\le|U|^*|C|$ are both established. From $q(x_i,r)\oplus q(x_j,r)=q(x_i,r)\vee q(x_j,r)-q(x_i,r)\wedge q(x_j,r)$, we can get $0\le\sum_{r\in U}|q(x_i,r)\oplus q(x_j,r)|/|q(x,r)\vee q(z,r)|\le|U|^*|C|$. Divide both sides of the inequality by $|U|^*|C|$ and $0\le 1/|U|^*|C|\sum_{r\in U}|q(x_i,r)\oplus q(x_j,r)|/|q(x_i,r)\vee q(x_j,r)|\le 1$ can be obtained. That is the inequality $0\le d(Q(x_i,U),Q(x_j,U))\le 1$ is established.

Theorem 2. *For $\forall x\in X$, feature subset Z and U satisfy $U\subseteq Z\subseteq R$. Let $Q(x,Z)$ and $Q(x,U)$ be fuzzy granular arrays on feature set U and Z. Inequality $|Q(x,U)|\le|Q(x,U)|$ is established.*

Proof 2. According to equation (13), if $\forall r_i\in U$, then $Q(x,U)=\sum_{i=1}^{|U|}q(x,r_i)/r_i$. Due to $U\subseteq Z$ and $r_i\in Z$, $Q(x,Z)=\sum_{i=1}^{|Z|}q(x,r_i)/r_i$ is established. Because of $U\subseteq Z\subseteq R$, for $r\in U$, we have $r\in Z$ and $|U|\le|Z|$. That is, if $q(x,r)\in Q(x,U)$, then $q(x,r)\in Q(x,Z)$. In sum, inequality $|Q(x,U)|\le|Q(x,Z)|$ is also established.

Below we give an example to explain the granulation process and measurement.

Example 1. As shown in Table 1, let $X=\{x_1,x_2,x_3,x_4\}$, $R=\{r_1,r_2,r_3\}$, and $Y=\{+1,-1\}$ be instance set, feature set, and label set, respectively. $C=\{c_1,c_2\}$ represents the cluster center set and parameter is $\lambda=0.5$. The fuzzy granulation is as follows.

We take instance x_1 as an example. The similarities between x and $c1$, $c2$, and $c3$ on feature r_1, r_2, and r_3 are $d(x_1,c_1,r_1)=1-|0.3-0.15|=0.85$, $d(x_1,c_2,r_1)=1-|0.3-0.35|=0.95$, $d(x_1,c_1,r_2)=1-|0.1-0.25|=0.85$, $d(x_1,c_2,r_2)=1-|0.1-0.45|=0.65$, $d(x_1,c_1,r_3)=1-|0.1-0.15|=0.95$, and $d(x_1,c_2,r_3)=1-|0.1-0.65|=0.65$, respectively. According to equation (7), fuzzy granules generated by x_1 on feature r_1, r_2, and r_3 are $q(x_1,r_1)=0.85/c_1+0.95/c_2$, $q(x_1,r_2)=0.85/c_1+0.65/c_2$, and $q(x_1,r_3)=0.95/c_1+0.65/c_2$, respectively.

In the same way, fuzzy granules generated by x_2 on feature set R are $q(x_2,r_1)=0.95/c_1+0.85/c_2$, $q(x_2,r_2)=0.95/c_1+0.75/c_2$, and $q(x_2,r_3)=0.95/c_1+0.45/c_2$ respectively.

According to $q(x_1,r_1)\vee q(x_2,r_1)=\sum_{j=1}^{k}(a+b-ab-\min\{(1-\lambda),a,b\}/\max\{\lambda,1-a,1-b\})/c_j$, (here, when $j=1$, $a=d(x_1,c_1,r_1), b=d(x_2,c_1,r_1)), \lambda=0.5$; when $j=2$, $a=d(x_1,c_2,r_1), b=d(x_2,c_2,r_1), \lambda=0.5$), we have

$$\begin{aligned}q(x_1,r_1)\vee q(x_2,r_1)&=\sum_{j=1}^{k}\frac{a+b-ab-\min\{(1-\lambda),a,b\}/\max\{\lambda,1-a,1-b\}}{c_j}\\&=\frac{0.85+0.95-0.85*0.95-\min\{(1-0.5),0.85,0.95\}/\max\{0.5,1-0.85,1-0.95\}}{c_1}\\&\quad+\frac{0.95+0.85-0.95*0.85-\min\{(1-0.5),0.95,0.85\}/\max\{0.5,1-0.95,1-0.85\}}{c_2}\\&=\frac{0.985}{c_1}+\frac{0.985}{c_2},\end{aligned}$$

Table 1: Fuzzy granulation and measurement of similarity.

$X\|C\|R\|Y\|\lambda$	r_1	r_2	r_3	y	λ
x_1	0.3	0.1	0.1	+1	0.5
x_2	0.2	0.2	0.1	+1	0.5
x_3	0.4	0.5	0.1	−1	0.5
x_4	0.1	0.2	0.7	−1	0.5
c_1	0.15	0.25	0.15	—	—
c_2	0.35	0.45	0.65	—	—

$$q(x_1,r_1)\wedge q(x_2,r_1)=\sum_{j=1}^{k}\frac{ab/\max\{\lambda,a,b\}}{c_j}=\frac{0.85*0.95/\max\{0.5,0.85,0.95\}}{c_1}+\frac{0.95*0.85/\max\{0.5,0.95,0.85\}}{c_2}=\frac{0.85}{c_1}+\frac{0.85}{c_2},$$

$$q(x_1,r_1)\oplus q(x_2,r_1)=q(x_1,r_1)\vee q(x_2,r_1)-q(x_1,r_1)\wedge q(x_2,r_1))=\frac{0.985-0.85}{c_1}+\frac{0.985-0.85}{c_2}=\frac{0.135}{c_1}+\frac{0.135}{c_2},$$

$$\left|q(x_1,r_1)\oplus q(x_2,r_1)\right|=\left|q(x_1,r_1)\wedge q(x_2,r_1)-q(x_1,r_1)\vee q(x_2,r_1)\right|=\left|\frac{0.985-0.85}{c_1}+\frac{0.985-0.85}{c_2}\right|=0.27. \tag{21}$$

Similarly, we also obtain

$$\begin{aligned}\left|q(x_1,r_2)\oplus q(x_2,r_2)\right|&=\left|\frac{0.135}{c_1}+\frac{0.175}{c_2}\right|\\&=0.31,\left|q(x_1,r_3)\oplus q(x_2,r_3)\right|\\&=\left|\frac{0.95}{c_1}+\frac{0.405}{c_2}\right|=1.355.\end{aligned} \tag{22}$$

Hence, the distance between instance x_1 and x_2 on feature set R with $\lambda = 0.5$ is

$$\begin{aligned}d(Q(x_1,R),Q(x_2,R))&=\frac{1}{|R|^*|C|}\sum_{r\in R}\frac{\left|q(x_1,r)\oplus q(x_2,r)\right|}{\left|q(x_1,r)\vee q(x_2,r)\right|}\\&=\frac{0.27+0.31+1.355}{3*2}=0.3225.\end{aligned} \tag{23}$$

4.3. Random Fuzzy Granular Decision Tree. RFGDT can embody structure and express the course of classifying instances on the basis of features. It includes a series of if-then rules, or it can also be regarded as a conditional probability distribution calculated in fuzzy granular space and label space. The strength is that this algorithm is readable and its efficiency is high. When learning, we employ the training data to generate a RFGDT on the basis of minimizing cost function. When predicting, test data are classified via the model. There are three steps in learning of RFGDT, namely, selecting feature, constructing tree, and pruning tree.

RFGDT is to describe the feature structure for classifying fuzzy granules in the fuzzy granular space, which can be composed of directed edges and nodes. A feature can be expressed by an internal node, and a label can be denoted by a leaf node.

During the classified process, the model starts from the root node, tests a certain fuzzy granule of instance, and assigns the fuzzy granule to its child nodes according to the result. Meanwhile, the value of a feature corresponds to each subnode. In this way, fuzzy granules are tested and allocated recursively until they reach the leaf node. Finally, fuzzy granules are allocated to the label of the leaf node.

Now, the definition of the fuzzy granular rule set is written as follows.

Definition 1. Suppose that $S = \{X, R, C, Y\}$ be a decision system, where $X = \{x_1, x_2, \ldots, x_n\}$ is an instance set, $R = \{r_1, r_2, \ldots, r_m\}$ is a feature set, $Y = \{y_1, y_2, \ldots, y_l\}$ is a label set, and $C = \{c_1, c_2, \ldots, c_K\}$ is a cluster center set. For $\forall x \in X$, a fuzzy granular array $Q(x, R)$ can be generated by instance x. Then, fuzzy granular space can be generated by $Q(x, R)$, that is,

$$H = W\wedge Q(x,R)\vee G, \tag{24}$$

where $W, G \in \mathbb{R}^{K\times m}$ are fuzzy granular array coefficients.

Suppose that $\forall x \in X$, $y \in Y$, a rule $\text{rule}_R(x) = [Q(R, x), y]$ can be made up of fuzzy granular array and label. Thus, a rule set $\text{RULE}_R = \{\text{rule}_R(x) | \forall x \in X\}$ can be constructed.

4.3.1. IF-Then Rule. RFGDT can also be regarded as a set of if-then rules. A RFGDT can be converted into an if-then rule set like this: A rule is constructed for every path from root

node to leaf node; the features of the internal nodes with regard to conditions of the rule, and the label of leaf node, correspond to the conclusion of rule. The path of RFGDT (corresponding to if-then rule set) has a key character: mutually exclusive and complete. This means that every fuzzy granular array is covered by a unique path or rule. The so-called coverage here means that the features of the fuzzy granular array are consistent with the features on the path.

4.3.2. Learning. The learning of RFGDT is to generalize a series of classification rules from training data. There may be more than one RFGDT (i.e., a RFGDT that can correctly classify the training data) that is not inconsistent with the training data. Our purpose is to find a RFGDT that has little contradiction with training data and has strong generalization ability. In other words, RFGDT learning is to estimate conditional probability on training data. Infinite conditional probability models exist by fuzzy granular space division. The conditional probability model chosen can not only have a great fitting to training data but also have a perfect prediction on test data. RFGDT learning uses a cost function to express the aim. As mentioned below, the cost function of RFGDT learning is usually a regularized maximum likelihood function, and its strategy is to minimize the cost function.

The algorithm of RFGDT learning is to recursively select the optimal feature and segment the training data on the basis of the feature, in order that each subdataset can get the best classified results. This process corresponds to the division of fuzzy granular space and the form of RFGDTs. At the beginning, the root node is constructed and all fuzzy granular arrays are placed at the root node. The algorithm chooses an optimal feature, and divides the training data into several subsets on the basis of this feature, in order that each subset has the best classification under the current conditions. If these subsets have been correctly classified, then the algorithm constructs leaf nodes and divides these subsets into the corresponding leaf nodes; if there are still subsets that cannot be correctly classified, then the algorithm selects new optimal features for these subsets, continues to segment them, constructs corresponding nodes, and proceeds recursively until all training data subsets are basically classified correctly or there is no suitable feature. Finally, each subset is divided into leaf nodes, i.e., there are clear categories. This generates a RFGDT.

The RFGDT produced may have better classification ability for training data but may have worse classification ability for test data, that is, overfitting may occur. We need to prune the tree from bottom to top to let the tree be simpler in order that it can enhance its generalization ability. Specifically, it is to remove the leaf nodes that are too subdivided, make them fall back to the parent node, or even an ancestor node, and then modify the parent node or an ancestor node to a new leaf node.

If the quantity of features is large, the features can also be chosen at the beginning of the RFGDT learning, leaving only features that have sufficient classification ability for training data. We can draw a conclusion that the learning algorithm includes feature selection, RFGDT construction, and RFGDT pruning. Since RFGDT denotes conditional probability distribution, RFGDTs of different depths correspond to probability models of different complexity. The generation of RFGDT corresponds to local selection of the model, and the pruning of RFGDT is related to global selection of the model. The generation of the RFGDT only considers the local optimum, while the pruning of the RFGDT considers the global optimum.

4.3.3. Feature Selection and Cost Function Construction. Selecting important features can enhance the efficiency of RFGDT learning. If the result of using a feature for classification is not very different from the result of random classification, the feature is said to have no classification ability. Empirically, throwing away such features has little effect on the accuracy. Here, we redefine the information gain ratio and use this criterion as the cost function of constructing a RFGDT. First, the empirical entropy of the dataset X is defined by

$$E(\mathbb{Q}) = -\sum_{i=1}^{l} \frac{|\mathbb{Q}_i|}{|\mathbb{Q}|} \log_2 \frac{|\mathbb{Q}_i|}{|\mathbb{Q}|}, \tag{25}$$

where $\mathbb{Q}$ denotes the set composed of fuzzy granular arrays, $|\mathbb{Q}|$ expresses the quantity of elements of the set, $\mathbb{Q}_i$ is the subset composed of fuzzy granular arrays of which classification is y_i, and $|\mathbb{Q}_i|$ is the quantity of elements of the set. Stipulate $0 \log_2(0) = 0$. It can be seen from the definition that entropy only depends on the distribution of (Q) and has nothing to do with the value of (Q). The greater the entropy is, the greater the uncertainty of the random variable is. $0 \le E(\mathbb{Q}) \le \log_2(l)$ can be verified from the definition.

The empirical conditional entropy of feature r on the fuzzy granular array set $\mathbb{Q}$ is

$$E(\mathbb{Q}|r) = \sum_{j=1}^{t} \frac{|\mathbb{Q}_j|}{|\mathbb{Q}|} E\left(\mathbb{Q}_j\right) = -\sum_{j=1}^{t} \frac{|\mathbb{Q}_j|}{|\mathbb{Q}|} \sum_{i=1}^{l} \frac{|\mathbb{Q}_{ji}|}{|\mathbb{Q}_i|} \log_2 \frac{|\mathbb{Q}_{ji}|}{|\mathbb{Q}_i|}, \tag{26}$$

where $\mathbb{Q}_j$ is the subset composed of fuzzy granular arrays taking the value r_j on the feature r, $|\mathbb{Q}_j|$ is the quantity of elements of the subset $\mathbb{Q}_j$, $\mathbb{Q}_{ji}$ is the subset composed of fuzzy granular arrays that takes the value r_j on feature r and the label y_i, and $|\mathbb{Q}_{ji}|$ is the quantity of elements of the subset $\mathbb{Q}_{ji}$.

The information gain is calculated as follows:

$$\delta(\mathbb{Q}, r) = E(\mathbb{Q}) - E(\mathbb{Q} \mid r). \tag{27}$$

We can now write the ratio $\widehat{\delta}(\mathbb{Q}, r)$ of information gain as

$$\widehat{\delta}(\mathbb{Q}, r) = \frac{\delta(\mathbb{Q}, r)}{E_r(\mathbb{Q})}. \tag{28}$$

Here, $E_r(\mathbb{Q}) = -\sum_{j=1}^{t} |\mathbb{Q}_j|/|\mathbb{Q}| \log_2 |\mathbb{Q}_j|/|\mathbb{Q}|$ (t denotes the quantity of taking the value on feature r).

4.3.4. RFGDT Generation. We adopt the information gain ratio criterion to select features and recursively build a RFGDT. The specific method is as follows.

Information gain ratio for each feature of each subdataset is calculated in the Map stage. Then, in the Reduce stage, the information gain ratios on the corresponding features of each subdataset are summed. The feature with the largest sum of information gain ratio can be chosen as the feature of the node, and the child node is constructed from the different feature values. We call the above approach recursively on the child nodes to build a RFGDT until the information gain ratios of all features are very small or there are no features to choose from. The algorithm is as follows:

Step I: $\mathbb{Q}$ is randomly divided into s subfuzzy granular array sets $\mathbb{Q}^{(j)}$, $j = 1, 2, \ldots, s$.

Step II: in the Map phase, the Map function uses each feature as the key and the information gain ratio as the value, namely, $[\text{key}, \text{value}] = [r, \widehat{\delta}(\mathbb{Q}^{(j)}, r)]$.

Step III: in the Reduce phase, Hadoop distributed system first aggregates the output results of all Map functions according to the key, and then uses these aggregated intermediate results as the input to the Reduce phase. The intermediate results after aggregation are as follows:

$$[\text{key}, \text{value}] = \left[r_k, \text{List}\left(\widehat{\delta}\left(\mathbb{Q}^{(j)}, r\right), \quad k = 1, 2, \ldots, s\right)\right]. \tag{29}$$

Step IV: if all fuzzy granular arrays in $\mathbb{Q}$ belong to the same class y_i, the algorithm sets T as a single-node tree, adopts y_i as the label of the node, and returns T.

Step V: if $R = \phi$, set T as a single-node tree, use the label y_i with the largest number of fuzzy granular arrays in $\mathbb{Q}$ as the class of the node, and return T.

Step VI: or else, calculate the sum of the information gain ratio of each feature in R to $\mathbb{Q}$ according to equation (28), and select the feature r^* with the largest sum of information gain ratio as the split feature, which can be written as

$$r^* = \arg\max_{1 \le j \le m} \sum_{k=1}^{s} \widehat{\delta}\left(\mathbb{Q}^{(k)}, r_j\right). \tag{30}$$

Step VII: if the information gain ratio of r^* is less than the threshold ϵ, set T as a single-node tree, and use the class y_i with the largest number of fuzzy granular arrays in $\mathbb{Q}$ as the label of the node, and return T.

Step VIII: if not, for each possible value v_j of r^*, according to $r = f_j$, divide $\mathbb{Q}$ into a number of nonempty subsets $\mathbb{Q}_i$, the label with the largest number of fuzzy granular arrays in $\mathbb{Q}_i$ is used as a mark, and the subnodes are constructed. The tree T is formed by the nodes and their subnodes, and return T.

Step IX: for the node i, use $\mathbb{Q}_i$ as training set and $R - \{r\}$ as feature set, recursively call Step I to Step VIII, get subtree T_i, and return T_i.

The algorithm is described as in Algorithm 2.

4.3.5. Pruning. The algorithm recursively generates RFGDT until it cannot do further. The tree generated in this way is often very accurate for the classification of training data, but the classification of test data is not so accurate, i.e., overfitting occurs. The main reason is that too much consideration is given to how to improve the correct classification of training data, thereby building an overly complex tree. The solution to this problem is to reduce tree complexity and simplify the constructed tree. The process of simplifying the constructed tree is called pruning. Specifically, pruning cuts some subtrees or leaf nodes from the constructed tree, and uses its root node or parent node as new leaf nodes, thereby simplifying the classification tree. The pruning of RFGDT can be achieved by minimizing the overall cost function of the tree.

Suppose that the quantity of leaf nodes of the tree T is $|T|$, t is the tree T leaf node, the leaf node has N_t fuzzy granular arrays, where there are N_{tk} fuzzy granular arrays in label y_k $(k = 1, 2, \ldots, l)$. $E_t(T)$ is the empirical entropy on the leaf node t, α is the parameter $(\alpha \ge 0)$, then the cost function of RFGDT learning can be written as

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t E_t(T) + \alpha|T|, \tag{31}$$

where $E_t(T) = -\sum_k N_{tk}/N_t \log_2 N_{tk}/N_t$. We have

$$C_\alpha(T) = C(T) + \alpha|T|. \tag{32}$$

In equation (31), $C(T)$ denotes the prediction error of the algorithm on training data, i.e., the fit degree between the algorithm and the training data, $|T|$ expresses model complexity, and parameter $\alpha \ge 0$ controls the impact between them. The larger α chooses a simpler model (tree), and the smaller one chooses a more complex model. $\alpha = 0$ means that only the fit between the model and the training data is considered, and the complexity of the model is not considered. Pruning is to select the algorithm with the smallest cost function when α is determined, i.e., the subtree with the smallest cost function. If α is determined, the larger the subtree is, the better the fit is to training data, but the higher the complexity is; on the contrary, the smaller the subtree, the lower the model complexity, but it often does not fit well to the training data. The cost function just shows the balance between the two. RFGDT generation only considers the better fit of the training data by increasing the information gain. The RFGDT pruning also considers the reduction of model complexity by optimizing the cost function. The RFGDT generation is a local learning model, and the RFGDT pruning is a global learning model. The following is a RFGDT pruning algorithm.

Step I: empirical entropy of each node is calculated.

Step II: recursively retract upward from the leaf nodes of the tree. Suppose that the whole tree before and after a group of leaf nodes retract to its parent node is T_b and T_a, and the corresponding cost function values are $C_\alpha(T_b)$ and $C_\alpha(T_a)$, respectively. If

$$C_\alpha(T_a) \le C_\alpha(T_b), \tag{33}$$

Input: instance set X and threshold ε
Output: root T of fuzzy granular decision tree
(1) Normalize instances into $[0, 1]$.
(2) Calculate cluster center set $C^* = \{c_1, c_2, \ldots, c_k\}$ (see Algorithm 1).
(3) $X_1 \cup X_2 \cup \ldots \cup X_p = X$and $\forall X_s, X_t \in X, X_s \cap X_t = \phi$//Parallel distributed fuzzy granulation.
(4) $\forall X_t \subset X$//This is parallel process. Here, take X_t as example.
For $i = 1$ to $|X_t|$
$\exists x_i \in X_t$
For $j = 1$ to M
$\exists r \in R$, instance x is fuzzy granulated as $q(x_i, r) = \sum_{j=1}^{k} d(x_i, c_j, r)/c_j$
end for
Build a fuzzy granular array $Q(x, R) = \sum_{j=1}^{|R|} q(x, r_j)/r_j$;
Get label of x_i, y_{x_i};
A rule can be built. $r_R(x_i) = [Q(x_i, R), y_{x_i}]$;
End for
(5) $\mathbb{Q}$ is randomly divided into s sub fuzzy granular array sets $\mathbb{Q}^{(j)}$, $j = 1, 2, \ldots, s$.
(6) Map stage. In the Map function, each feature r is used as the key of the Map function, and the information gain ratio is used as the value of the Map function, namely, [key, value] = $[r, \widehat{\delta}(\mathbb{Q}^{(j)}, r)]$
(7) IF for $\forall Q_i \in \mathbb{Q}$, its classification is y_i, then set T as a single-node tree, take y_i as the label of the node, and return T.
(8) Reduce stage. Between the Map phase and the Reduce phase, the Hadoop distributed system first aggregates the output results of all Map functions according to the key.
Then, these aggregated intermediate results are used as the input of the Reduce stage, and the intermediate results after aggregation are as follows. [key, value] = $[r_k, \text{List}(\widehat{\delta}(\mathbb{Q}^{(j)}, r), k = 1, 2, \ldots, s)]$
(9) If $R = \phi$, set T as a single-node tree, and use the label y_i with the largest number of fuzzy granular arrays in $\mathbb{Q}$ as the label of the node, and return T.
Or else, $\forall r \in R$, calculate the information gain ratio of r to $\mathbb{Q}^j$, namely $\widehat{\delta}(\mathbb{Q}^j, r) = \delta(\mathbb{Q}^j, r)/E_r(\mathbb{Q}^j)$, select the feature with the largest sum of information gain ratio $\widehat{\delta}(\mathbb{Q}^j, r) = \delta(\mathbb{Q}^j, r)/E_r(\mathbb{Q}^j)$.
(10) If the information gain of r satisfies $\widehat{\delta}(\mathbb{Q}, r) < \varepsilon$, then T is set as a single-node tree, and the label y_i with the largest number of fuzzy granular arrays in $\mathbb{Q}$ is used as the class of the node, and T is returned;
Or else, for each possible value v_j of r, according to $r = f_j$, divide $\mathbb{Q}$ into subsets of nonempty, $\mathbb{Q}_i$, take the label with the largest number of fuzzy granular arrays in $\mathbb{Q}_i$ as a mark, construct subnodes, and form tree T based on the nodes and their subnodes, and return T.
(11) For the node i, use $\mathbb{Q}_i$ as the training set and $R - \{r\}$ as the feature set, recursively call 5 to 10, and get the subtree T_i, and return T_i.

Algorithm 2: Construct RFGDT.

then pruning, that is, the parent node becomes a new leaf node.

Step III: go to Step II, until it cannot continue, and get the subtree with the smallest cost function T_α.

The algorithm is shown as in Algorithm 3.

4.3.6. Label Prediction. After the RFGDT is constructed, given a test instance, first we transform it into fuzzy granular array and then use the RFGDT decision tree trained to predict. The method is described further in Algorithm 4.

5. Experimental Analysis

This paper employs 4 datasets from the UC Irvine Machine Learning Repository as the data source for the experimental test and constructs 8 datasets with 1% noise and 3% noise, respectively, as demonstrated in Table 2. The tenfold cross-validation method was adopted in the experiments. Data of 90% and 70% were chosen randomly as training sets, respectively, and the remaining data were taken as the test set to execute one verification. Then, we repeated the process ten times. The running time and average accuracy were used as measurements of performance. As illustrated in Figure 2, serial clustering fuzzy granulation, parallel clustering fuzzy granulation proposed, and serial granulation were compared on efficiency. C4.5, Support Vector Machines (SVMs), Convolutional Neural Networks (CNN), and RFGDT were compared for average accuracy (see Figures 3–5). In RFGDT classifier, the quantity of cluster centers is a key parameter that has an effect on performance of classification, such as accuracy. The relation between the quantity of cluster centers and the average accuracy was analyzed.

Fuzzy granulation in the case of serial computing tasks has lower efficiency, which cannot meet the needs. If computing task is parallelizable, serial computing task can be converted into parallel computing one. This paper adopted the approach of parallel granulation via clustering. We usually employ MapReduce to deal with parallel tasks of large-scale data. Fuzzy granulation of large-scale dataset can be divided into several subcomputing tasks and the sub-datasets can be assigned to computing nodes via abstracting a hierarchical computing model. Due to its simple and easy-to-use programming interface, MapReduce has been widely used in parallel programming model and computing

Input: fuzzy granular decision tree T and parameter α
Output: pruned subtree T_α
(1) Calculate the empirical entropy of each node $E_t(T) = -\sum_k N_{tk}/N_t \log N_{tk}/N_t$
(2) Recursively retract upward from the leaf nodes of the tree.
Suppose that the whole tree before and after a group of leaf nodes retract to its parent node is T_b and T_a, and the corresponding cost function values are $C_\alpha(T_b)$ and $C_\alpha(T_a)$.
(3) If $C_\alpha(T_a) \leq C_\alpha(T_b)$, pruning is executed, that is, to change the parent node into a new leaf node.
(4) Return to Step 2 until it cannot continue, and get the subtree with the smallest cost function T_α

Algorithm 3: Pruning of RFGDT.

Input: test instance x, root node of RFGDT, and cluster center set C.
Output: label y_x of instance x
(1) Obtain fuzzy clustering granular vector $Q(x, R)$.
(2) Judge the path from the root node to the leaf node according to the feature value, and get the label according to the leaf node.

Algorithm 4: RFGDT prediction.

Table 2: Datasets from the UC Irvine Machine Learning Repository.

No.	Datasets	Number of instances	Number of features	Number of classifications
1	Wine Quality	4898	12	11
2	Bank Marketing	45211	17	2
3	Localization Data for Person Activity	164860	8	8
4	IDA2016Challenge	76000	171	2

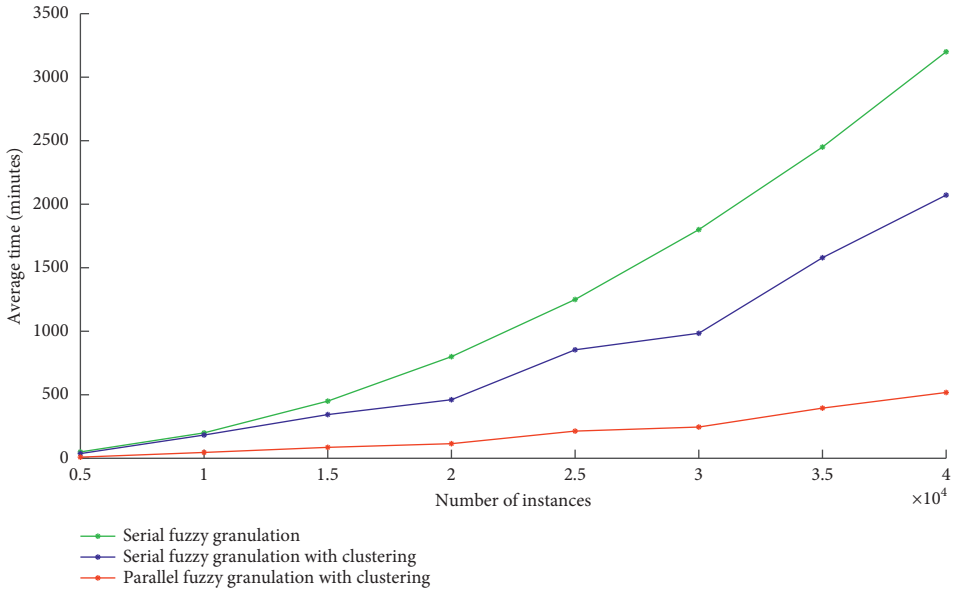

Figure 2: Comparison of executing time.

framework. The main thought is as follows. Job is divided into multiple independently runnable Map tasks; these Map tasks are distributed to several processors to execute; intermediate results are generated; the reduce operation tasks are combined to produce final output results. There are two parts in MapReduce calculation process, namely, Map and Reduce. The input [key, value] is received by Map (see Table 3), and then intermediate results ([key, value]) are output, where key = x_i, value = $[h(x_i, r_1), \ldots, h(x_i, r_m), C]$ (see Table 4).

The output of Reduce is like key = x_i, value = $[q(x_i, r_1), q(x_i, r_2), \ldots, q(x_i, r_m)]$ (see Table 5). Without clustering, the complexity of granulation is $O(n^2 * m)$; With clustering, the complexity of granulation time can reach $O(n * k * m)$ $(k < m)$; If fuzzy granulation is proceeded by the parallel method, the complexity can

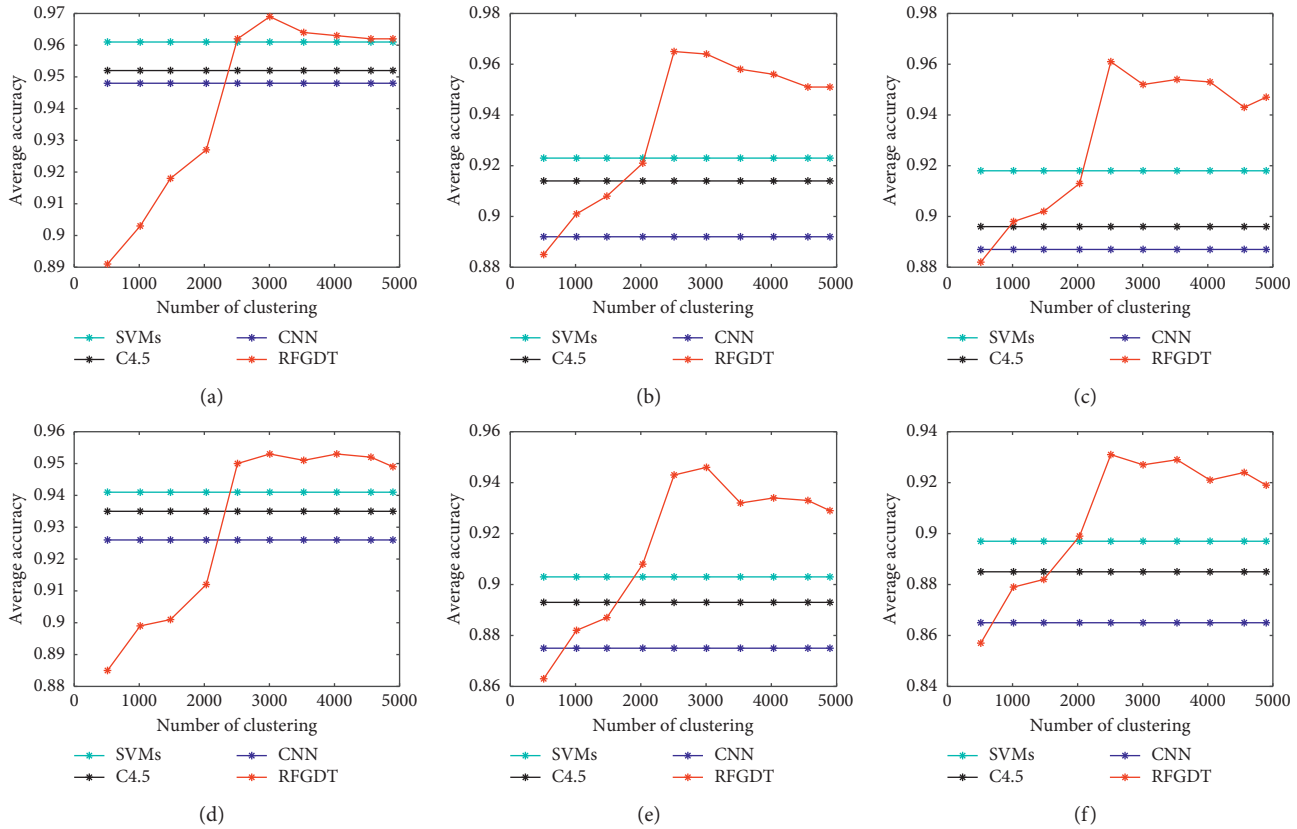

Figure 3: Comparison of dataset Wine Quality. (a) Data of 90% as the training set. (b) Data with noise of 1% and data of 90% as the training set. (c) Data with noise of 3% and data of 90% as the training set. (d) Data of 70% as the training set. (e) Data with noise of 1% and data of 70% as the training set. (f) Data with noise of 3% and data of 70% as the training set.

achieve $O(n * k * m/B)$, where B denotes the quantity of subsets, k represents the quantity of cluster centers, n expresses the quantity of instances, m is the quantity of features, and each subset corresponds to a Map.

Figure 2 compares the efficiency of traditional granulation, serial granulation with clustering, and the parallel granulation with clustering proposed, where the abscissa expresses the quantity of instances and the ordinate denotes the average time taken for granulation. Below n represents the quantity of instances and K is the quantity of cluster center. When $n = 5000$ and $K = 3692$, the serial granulation cost 50 mins, the running time of serial clustering granulation was 37 mins, and the running time of parallel clustering granulation was only 9 mins. The parallel clustering granulation reduced by 26.00% and 82.00%, respectively. When $n = 20,000$ and $K = 11,524$, the serial granulation took 800 mins, while the running time of the serial clustering granulation was 461 mins (i.e., 42.38% improvement). Parallel granulation with clustering executed only 115 mins and was enhanced by 85.63% and 75.05%, respectively. When $n = 30,000$ and $K = 16,398$, the running time of parallel granulation with clustering improved by 45.33% and 86.33%, respectively, compared with the other two methods. We can draw a conclusion that as the quantity of instances increases, serial clustering granulation and parallel clustering granulation methods increase the efficiency to a great extent.

Taking 90% of the dataset Wine Quality as the training set, we can get the following results. As shown in Figure 3(a), in the dataset Wine Quality, when the quantity of cluster centers was less than 3006, the average accuracy of RFGDT was lower than that of the other three methods. With the quantity of cluster centers rising, the average accuracy of RFGDT increases rapidly. Especially, when the quantity of cluster centers was 3006, it achieved a peak value of 0.969, while the average accuracies of SVMs, C4.5, and CNN were 0.961, 0.952, and 0.948 (i.e., 0.83%, 1.79%, and 2.22% improvement, respectively). When the quantity of cluster centers was greater than 3006, the average accuracy of RFGDT also decreased slightly, but it was still higher than the other three methods. After adding 1% noise in the data, as illustrated in Figure 3(b), when the quantity of cluster centers was 3006, the average accuracy of RFGDT reached a peak value of 0.964, while the average accuracies of SVMs, C4.5, and CNN, RFGDT were 0.923, 0.914, and 0.892, respectively. RFGDT improved by 4.44%, 5.47%, and 8.07% respectively. Comparing these two datasets, SVMs, C4.5, and CNN dropped by 3.95%, 3.99%, and 5.91% respectively, and RFGDT reduced by 0.52% at the peak value. It can be seen from statistical data that RFGDT is more robust and stable to noise. When we add 3% noise to the data, as exhibited in Figure 3(c), SVMs, C4.5, CNN, and RFGDT decreased by about 0.54%, 1.97%, 0.56%, and 0.42%, respectively, compared with noise data of 1%. After that, we took 70% of data

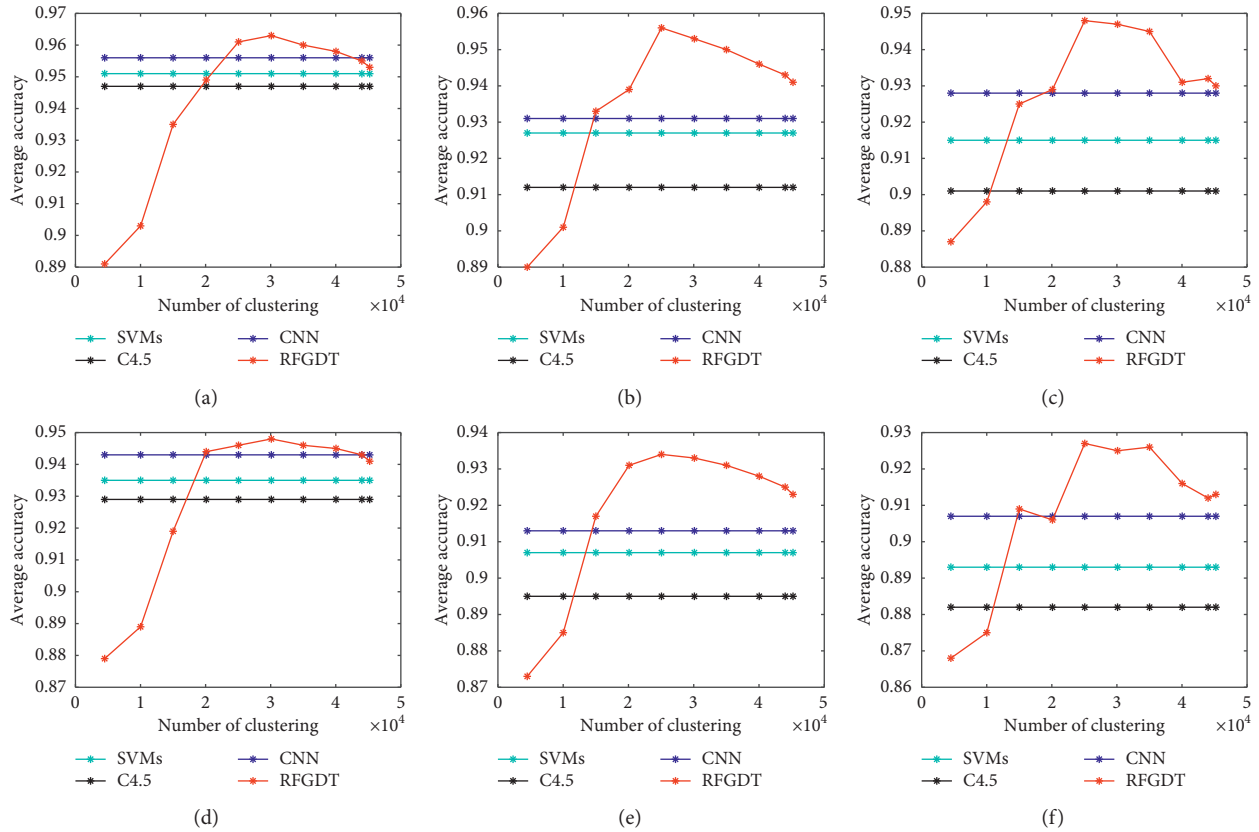

Figure 4: Comparison of dataset Bank Marketing. (a) Data of 90% as the training set. (b) Data with noise of 1% and data of 90% as the training set. (c) Data with noise of 3% and data of 90% as the training set. (d) Data of 70% as the training set. (e) Data with noise of 1% and data of 70% as the training set. (f) Data with noise of 3% and data of 70% as the training set.

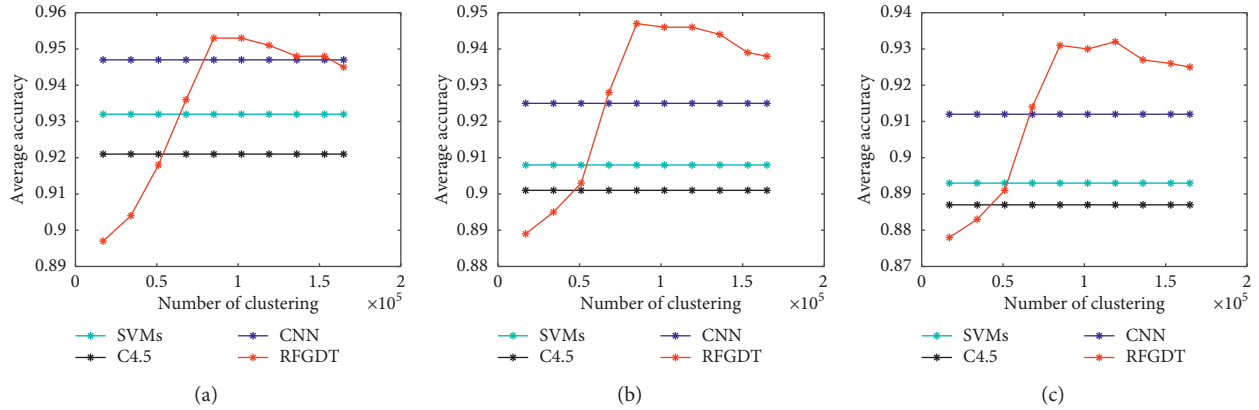

Figure 5: Continued.

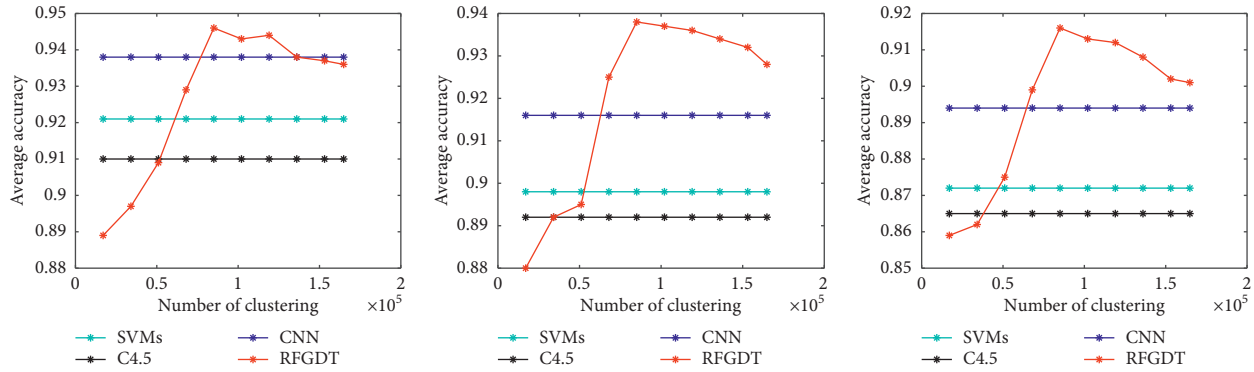

Figure 5: Comparison of dataset Localization Data for Person Activity. (a) Data of 90% as the training set. (b) Data with noise of 1% and data of 90% as the training set. (c) Data with noise of 3% and data of 90% as the training set. (d) Data of 70% as the training set. (e) Data with noise of 1% and data of 70% as the training set. (f) Data with noise of 3% and data of 70% as the training set.

Table 3: Map process input.

Task of map	Key	Value
Map 1	x_1	$[h(a_1, x_1), \ldots, h((a_m, x_1), C]$
	x_2	$[h(a_1, x_2), \ldots, h(a_m, x_2), C]$
	...	...
Map 2	x_i	$[h(a_1, x_i), \ldots, h(a_m, x_i), C]$
	x_{i+1}	$[h(a_1, x_{i+1}), \ldots, h(a_m, x_{i+1}), C]$
	...	...
Map...	...	...
Map B	x_j	$[h(a_1, x_j), \ldots, h(a_m, x_j), C]$
	x_{j+1}	$[h(a_1, x_{j+1}), \ldots, h(a_m, x_{j+1}), C]$
	...	...

Table 4: Map process output.

Task of map	Key	Value
Map 1	x_1	$[q(x_1, r_1), r_1]$
	x_1	$[q(x_1, r_2), r_2]$
	...	...
	x_1	$[q(x_1, r_m), r_m]$
	x_2	$[q(x_2, r_1), r_1]$
	x_2	$[q(x_2, r_2), r_2]$
	...	...
	x_2	$[q(x_2, r_m), r_m]$
	...	...
Map 2	x_i	$[q(x_i, r_1), r_1]$
	x_i	$[q(x_i, r_2), r_2]$
	...	...
	x_i	$[q(x_i, r_m), r_m]$
	...	...
Map...	...	...
Map B	x_j	$[q(x_j, r_1), r_1]$
	x_j	$[q(x_j, r_2), r_2]$
	...	...
	x_j	$[q(x_j, r_m), r_m]$
	...	...

as the training set to verify the performance. Overall, the average accuracies of these four methods were on the decline. From Figures 3(d)–3(f), RFGDT performs better than other three algorithms when the number of clustering is more than 2000.

Table 5: Reduce process output.

Key	Value
x_1	$[q(x_1, r_1), q(x_1, r_2), \ldots, q(x_1, r_m)]$
x_2	$[q(x_2, r_1), q(x_2, r_2), \ldots, q(x_2, r_m)]$
...	...
x_n	$[q(x_n, r_1), q(x_n, r_2), \ldots, q(x_n, r_m)]$

As illustrated in Figure 4(a), dataset BankMarketing contained nearly 50,000 instances, which was 10 times the scale of dataset Wine Quality. The shape of the average accuracy curve of RFGDT was similar to Figure 3, and the overall shape was high in the middle and low on both sides. When the quantity of cluster centers was $K = 30100$, the average accuracy of RFGDT reached the largest value of 0.963, while SVMs, C4.5, and CNN were 0.951, 0.947, and 0.955, respectively (i.e., 1.26%, 1.69%, and 0.84% improvement, respectively). In the Bank Marketing dataset with noise, as demonstrated in Figure 4(b), RFGDT reached a peak value of 0.956 at $K = 30100$. Compared with SVMs, C4.5, and CNN, RFGDT increased by 3.13%, 4.82%, and 2.69%, respectively. RFGDT reduced by 0.73%, while SVMs, C4.5, and CNN reduced by 2.59%, 3.84%, and 2.69%, respectively. As can be seen, RFGDT is not sensitive to noise and C4.5 is more sensitive to noise. Figure 4(c) shows the four algorithms are all reduced when the percent of noise data was 3%. However, RFGDT performed better than SVMs, C4.5, and CNN by about 3.61%, 5.22%, and 2.16% regarding the average accuracy, respectively. When 70% of data were taken as the training set, the four algorithms were decreased compared with 90% of data being the training set. However, as shown in Figures 4(d)–4(f), RFGDT outperforms the other three algorithms under most parameters.

The quantity of instances in dataset Localization Data for Person Activity was more than 160,000. As illustrated in Figure 5(a), without noise, when $K = 85060$, the average accuracy curve of RFGDT reached a peak value of 0.953, while SVMs was 0.932, C4.5 was 0.922, and CNN was 0.947 (2.25%, 3.36%, and 0.63% improvement, respectively). CNN

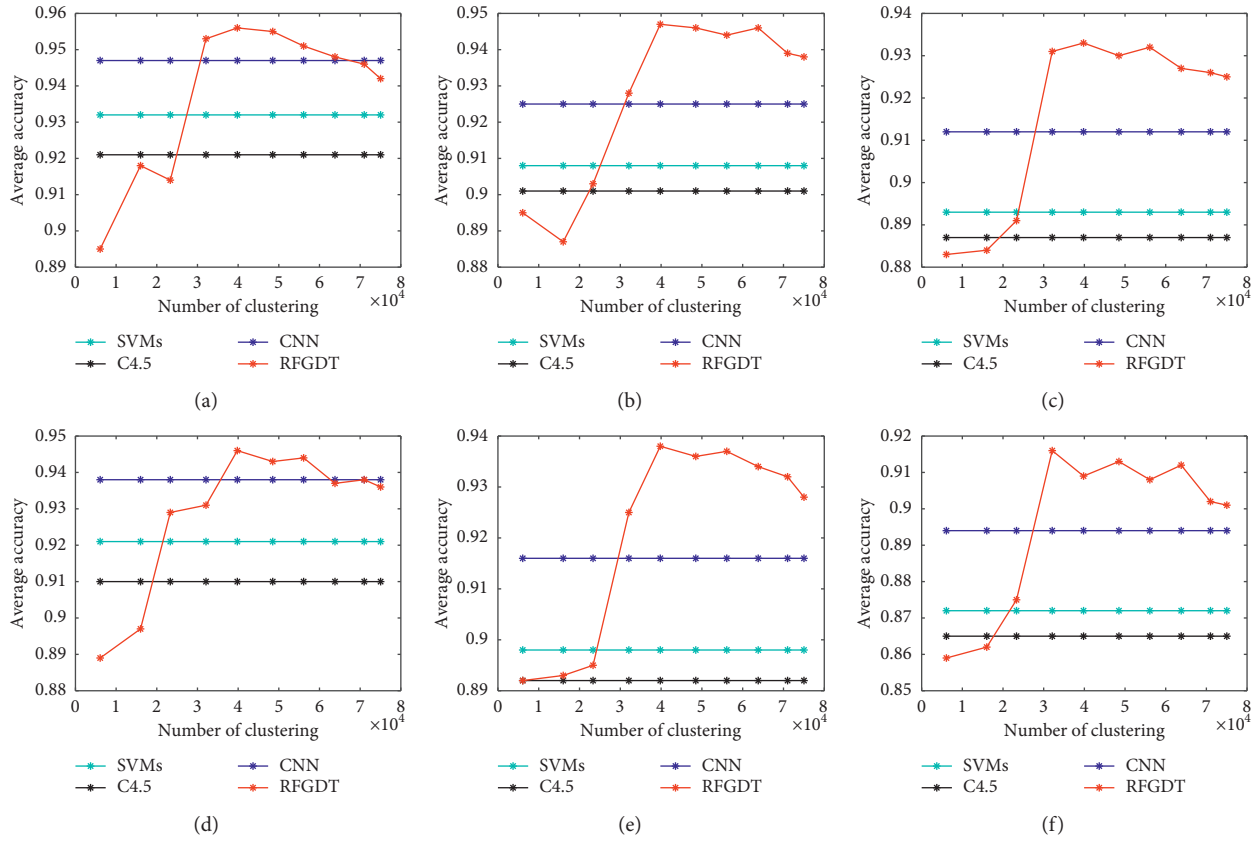

Figure 6: Comparison of dataset IDA2016Challenge. (a) Data of 90% as the training set. (b) Data with noise of 1% and data of 90% as the training set. (c) Data with noise of 3% and data of 90% as the training set. (d) Data of 70% as the training set. (e) Data with noise of 1% and data of 70% as the training set. (f) Data with noise of 3% and data of 70% as the training set.

performs better than SVMs, while SVMs is better than C4.5, and RFGDT is slightly better than CNN. In the dataset with noise, as demonstrated in Figure 5(b), compared with SVMs, C4.5, and CNN, the peak value of RFGDT increased by 4.30%, 4.86%, and 2.38%, respectively. Compared with SVMs, C4.5, RFGDT and CNN are less sensitive to noise. As shown in Figure 5(c), when noise occupied 3% of data, the four algorithms were all decreased. But RFGDT achieved 0.932 of average accuracy under $K = 119{,}031$, while SVMs, C4.5, and CNN just got 0.893, 0.887, and 0.912, respectively (i.e., 4.37%, 5.07%, and 2.19% improvement). Figures 5(d)–5(f) compare the performance with 70% of data being the training set and show that RFGDT performs better than SVMs, C4.5, and CNN under special parameters.

The dimension in dataset IDA2016Challenge is much higher than the other three datasets. We took 90% and 70% of data to test as training sets, respectively. On the basis of this, we added 1% and 3% noise into the dataset, respectively. The detailed results are as follows. As shown in Figure 6(a), when $K = 39800$, RFGDT achieved a peak value of 0.956, while SVMs, C4.5, and CNN just got 0.932, 0.921, and 0.947, respectively (i.e., about 2.58%, 3.80%, and 0.95% improvement, respectively). After adding noise of 1%, as illustrated in Figure 6(b), the highest value of RFGDT was 0.947 and RFGDT increased by about 4.30%, 5.11%, and 2.38% compared with SVMs, C4.5, and CNN, respectively. After adding noise of 3%, as demonstrated in Figure 6(c), the four algorithms were all decreased, but RFGDT was better than the other three algorithms. When data of 70% is taken as training set, RFGDT still outperforms SVMs, C4.5, and CNN, as exhibited in Figures 6(d)–6(f).

Besides the dataset mentioned above, we also applied the algorithm to predict Alzheimer's disease by voice. This dataset was from the University of Pittsburgh and was stored in the form of speech and text from participants containing elderly controls, people with possible Alzheimer's Disease, and people with other dementia diagnoses. The corpus included 1263 instances. Mel-frequency Cepstral Coefficients (MFCC) of corpus was extracted as features for prediction. We calculated the first 20 dimensions, their first-order difference, and their second-order difference, which were concatenated to get 57-dimensional features. The precision of RPFDT was a maximum of 0.932. We can predict Alzheimer's disease by voice, which is a simple and low cost method compared with Magnetic Resonance Imaging. It is very meaningful and valuable for the diagnosis of Alzheimer's disease.

From the above analysis, we can see that the average accuracy of RFGDT is better than SVMs, C4.5, and CNN in the six datasets. In smaller datasets, CNN performs weaker than SVMs and C4.5. Especially, in datasets containing noise, the average accuracy of RFGDT is stable

and less sensitive to noise. Judging from the curve shape of the average accuracy of RFGDT, it shows a form of high and low in the middle. When the value of K is small, the performance of RFGDT is weaker than the other three algorithms. As the value of K increases, the performance of RFGDT is better than the other three algorithms. We use 10-fold cross-validation, and the test set and training set are obtained randomly. In other words, for each time, the algorithm is evaluated in different training set and test set; it has randomness, but the performance of the algorithm is just evaluated objectively. The imbalance of instances will also affect the performance of the algorithm. For noisy datasets, we also found that RFGDT is more robust. The main reason lies in the fuzzy granulation process. RFGDT embodies a global comparison thought, which can overcome the noise interference to some extent. This is also the advantage of the RPFDT. At the same time, we also found that the choice of the K value is also the key to classification. If the K value is too small, it will reduce the classification accuracy. Instead, if the K value is too large, it will increase noise and also reduce the classification effect. A reasonable value of K is also the key to the performance of the algorithm. Compared with classical methods, granulation process costs some time, but this process can be executed offline. Moreover, parallel granulation can improve the efficiency greatly. In the meantime, in the granular space, the accuracy of classification can be enhanced.

6. Discussion

In this study, we propose a RFGDT that is suitable for dual-classification or multiclassification problems. In the algorithm, the idea of parallel distributed granulation is introduced, which improves the efficiency of data granulation. In the parallel granulation process, we design AGRC for granulation. We transform a classified problem of data into fuzzy granular space to find the solution. In the fuzzy granular space, we define fuzzy granules, fuzzy granular arrays on the basis of operators designed. The aim is to use the information gain rate to select feature as the split point to recursively construct the fuzzy granular decision tree. In order to avoid overfitting, we also design the pruning algorithm of RFGDT, which can improve the performance further. In the future, we will apply it to cloud computing and big data.

Data Availability

The dataset used to support the findings of this study is from the UC Irvine Machine Learning Repository.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Scientific Research "Climbing" Program of Xiamen University of Technology under Grant nos. XPDKT20027 and XPDKQ18011, in part by the National Natural Science Foundation of China under Grant nos. 61976183 and 41804019, in part by the Natural Science Foundation of Fujian Province of China under Grant nos. 2019J01850 and 2018J01480, in part by Project of Industry-University-Research of Xiamen University and Scientific Research Institute under Grant no. 3502Z20203064, in part by University Natural Sciences Research Project of Anhui Province of China under Grant no. KJ2020A0660, and in part by the Natural Science Foundation of Anhui Province of China under Grant no. 2008085MF202.

References

[1] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[2] J. C. Schlimmer and D. Fisher, "A case study of incremental concept induction," in *Proceedings of the 1986 National Conference on Artificial Intelligence*, pp. 496–501, Philadelphia, PA, USA, August 1986.

[3] P. E. Utgoff, "Id5: An incremental id3," in *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, USA, June 1988.

[4] X. Liu and S. Li, "An optimized algorithm of decision tree," *Journal of Software*, vol. 9, no. 10, pp. 797–800, 1998, (in Chinese).

[5] X. Wang and C. Yang, "Merging-branches impact on decision tree induction," *Chinese Journal of Computers*, vol. 30, no. 8, pp. 1251–1258, 2007, (in Chinese).

[6] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[7] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, USA, August 2000.

[8] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, August 2001.

[9] J. Su and H. Zhang, "A fast decision tree learning algorithm," in *Procedings of 21th National Conference on Artificial Intelligence*, pp. 500–505, Boston, MA, USA, July 2006.

[10] A. E. Khedr, A. M. Idrees, and A. I. El Seddawy, "Enhancing iterative dichotomiser 3 algorithm for classification decision tree," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 70–79, 2016.

[11] A. Franco-Arcega, J. A. Carrasco-Ochoa, G. Sánchez-Díaz, and J. Fco Martínez-Trinidad, "Building fast decision trees from large training sets," *Intelligent Data Analysis*, vol. 16, no. 4, pp. 649–664, 2012.

[12] C. Li, Y. Zhang, and X. L. Ocvfdt, "One-class very fast decision tree for one-class classification of data stream," in *Proceedings of the 3rd International Workshop on Knowledge Discovery from Sensor Data*, Paris, France, June 2009.

[13] G. Liu, H. Cheng, Z. Qin, and Q. Liu, "E-cvfdt: an improving cvfdt method for concept drift data stream," in *Proceedings of the 2013 International conference on Communications, Circuits and Systems*, pp. 315–318, Chengdu, China, November 2013.

[14] M. Batra and R. Agrawal, "Comparative analysis of decision tree algorithms," *Nature Inspired Computing*, Springer, Singapore, pp. 31–36, 2018.

[15] A. Cherfi, K. Nouira, and A. Ferchichi, "Very fast c4.5 decision tree algorithm," *Applied Artificial Intelligence*, vol. 32, no. 2, pp. 119–137, 2018.

[16] X. Dong, M. Qian, and R. Jiang, "Packet classification based on the decision tree with information entropy," *The Journal of Supercomputing*, vol. 76, no. 6, pp. 4117–4131, 2020.

[17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, USA, 1984.

[18] M. Mehta, R. Agrawal, and J. Rissanen, "SLIQ: a fast scalable classifier for data mining," in *Proceedings of the 1996 International Conference on Extending Database Technology*, Avignon, France, March 1996.

[19] B. Chandra and P. Pallath, "On improving efficiency of SLIQ decision tree algorithm," in *Proceedings of the 2007 International Joint Conference on Neural Networks*, Orlando, FL, USA, August 2007.

[20] R. Rastogi and K. Shim, "Public: a decision tree classifier that integrates building and pruning," *Data Mining and Knowledge Discovery*, vol. 4, no. 4, pp. 315–344, 2000.

[21] J. Gehrke, R. Ramakrishnan, and V. Ganti, "Rainforest-a framework for fast decision tree construction of large datasets," *Data Mining and Knowledge Discovery*, vol. 4, no. 2/3, pp. 127–162, 2000.

[22] D. Miao and J. Wang, "Rough sets based approach for multivariate decision tree construction," *Journal of Software*, vol. 6, pp. 425–431, 1997, (in Chinese).

[23] X.-Z. Wang, J.-H. Zhai, and S.-X. Lu, "Induction of multiple fuzzy decision trees based on rough set technique," *Information Sciences*, vol. 178, no. 16, pp. 3188–3202, 2008.

[24] J.-h. Zhai, "Fuzzy decision tree based on fuzzy-rough technique," *Soft Computing*, vol. 15, no. 6, pp. 1087–1096, 2011.

[25] J.-M. Wei, S.-Q. Wang, M.-Y. Wang, J.-P. You, and D.-Y. Liu, "Rough set based approach for inducing decision trees," *Knowledge-Based Systems*, vol. 20, no. 8, pp. 695–702, 2007.

[26] F. Jiang, Y. Sui, and C. Cao, "An incremental decision tree algorithm based on rough sets and its application in intrusion detection," *Artificial Intelligence Review*, vol. 40, no. 4, pp. 517–530, 2013.

[27] Q. Hu, X. Che, L. Zhang, and D. Yu, "Rand entropy-based decision tress for monotonic classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 11, pp. 2052–2064, 2013.

[28] Y. Qian, H. Xu, J. Liang, B. Liu, and J. Wang, "Fusing monotonic decision trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 10, pp. 2717–2728, 2015.

[29] S. Pei, Q. Hu, and C. Chen, "Multivariate decision trees with monotonicity constraints," *Knowledge-Based Systems*, vol. 112, pp. 14–25, 2016.

[30] J. C. Shafer, R. Agrawal, and M. Mehta, "Sprint: a scalable parallel classifier for data mining," in *Proceedings of the 1996 22th International Conference on Very Large Data Bases*, pp. 544–555, Mumbai, India, September 1996.

[31] R. Kufrin, "Decision trees on parallel processors," *Machine Intelligence and Pattern Recognition*, vol. 20, pp. 279–306, 1997.

[32] M. V. Joshi, G. Karypis, and V. Kumar, "Scalparc: a new scalable and efficient parallel classification algorithm for mining large datasets," in *Proceedings of the 1998 First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pp. 573–579, Orlando, FL, USA, March 1998.

[33] A. Srivastava, E.-H. Han, V. Kumar, and V. Singh, "Parallel formulations of decision-tree classification algorithms," *Data Mining and Knowledge Discovery*, vol. 3, no. 3, pp. 237–261, 1999.

[34] Y. Sheng, V. V. Phoha, and S. M. Rovnyak, "A parallel decision tree-based method for user authentication based on keystroke patterns," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 4, pp. 826–833, 2005.

[35] B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo, "Planet," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1426–1437, 2009.

[36] K. Walkowiak and M. Wozniak, "Decision tree induction methods for distributed environment," *Man-Machine Interactions*, pp. 201–208, Springer, Heilderberg, Germany, 2009.

[37] W. Yin, Y. Simmhan, and V. K. Prasanna, "Scalable regression tree learning on hadoop using openplanet," in *Proceedings of the 2012 3rd International Workshop on MapReduce and Its Applications, MapReduce '12*, Delft, The Netherlands, June 2012.

[38] R. Wang, Y.-L. He, C.-Y. Chow, F.-F. Ou, and J. Zhang, "Learning elm-tree from big data based on uncertainty reduction," *Fuzzy Sets and Systems*, vol. 258, pp. 79–100, 2015.

[39] Y. Mu, X. Liu, Z. Yang, and X. Lin, "A parallel c4.5 decision tree algorithm based on mapreduce," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 8, pp. 1–12, 2017.

[40] Y. Mu, L. Wang, and X. Liu, "A fast rank mutual information based decision tree and its implementation via map-reduce," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 10, pp. 1–22, 2018.

[41] A. Segatori, F. Marcelloni, and W. Pedrycz, "On distributed fuzzy decision trees for big data," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 1, pp. 174–192, 2018.

[42] Y. Mu, X. Liu, and L. Wang, "A Pearson's correlation coefficient based decision tree and its parallel implementation," *Information Sciences*, vol. 435, pp. 40–58, 2018.

[43] W. Li, Y. Chen, H. Hu, and C. Tang, "Using granule to search privacy preserving voice in home iot systems," *IEEE Access*, vol. 8, pp. 31957–31969, 2020.

[44] W. Li, Z. Wei, Y. Chen, C. Tang, and Y. Song, "Fuzzy granular hyperplane classifiers," *IEEE Access*, vol. 8, pp. 112066–112077, 2020.

[45] W. Li, J. Xue, Y. Chen et al., "Fuzzy granule manifold alignment preserving local topology," *IEEE Access*, vol. 8, pp. 178695–178705, 2020.

[46] W. Li, K. Zhang, Y. Chen, C. Tang, X. Ma, and Y. Luo, "Random fuzzy clustering granular hyperplane classifier," *IEEE Access*, vol. 8, pp. 228767–228778, 2020.

[47] W. Li, Y. Chen, and Y. Song, "Boosted k-nearest neighbor classifiers based on fuzzy granules," *Knowledge-Based Systems*, vol. 195, pp. 1–13, 2020.

[48] C. Fu, W. Lu, W. Pedrycz, and J. Yang, "Rule-based granular classification: a hypersphere information granule-based method," *Knowledge-Based Systems*, vol. 194, pp. 1–16, 2020.

[49] P. Singh, Y.-P. Huang, and S.-I. Wu, "An intuitionistic fuzzy set approach for multi-attribute information classification and decision-making," *International Journal of Fuzzy Systems*, vol. 22, no. 5, pp. 1506–1520, 2020.