International Journal of Computational Intelligence Systems
Vol. **14(1)**, 2021, *pp.* 257–265
DOI: https://doi.org/10.2991/ijcis.d.201202.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

Feature-Weighting and Clustering Random Forest

Zhenyu Liu[1,2,*], Tao Wen[1,2], Wei Sun[2], Qilong Zhang[1]

[1]*College of Computer Science and Engineering, Northeastern University, Shenyang, China*
[2]*Department of Computer Science and Technology, Dalian Neusoft University of Information, Dalian, China*

ARTICLE INFO

Article History
Received 05 Jul 2020
Accepted 27 Nov 2020

Keywords
Random forest
Feature weighting
Node split method
Categorical feature
Decision tree ensemble

ABSTRACT

Classical random forest (RF) is suitable for the classification and regression tasks of high-dimensional data. However, the performance of RF may be not satisfied in case of few features, because univariate split method cannot bring more diverse individuals. In this paper, a novel method of node split of the decision trees is proposed, which adopts feature-weighting and clustering. This method can combine multiple numerical features, multiple categorical features or multiple mixed features. Based on the framework of RF, we use this split method to construct decision trees. The ensemble of the decision trees is called Feature-Weighting and Clustering Random Forest (FWCRF). The experiments show that FWCRF can get the better ensemble accuracy compared with the classical RF based on univariate decision tree on low-dimensional data, because FWCRF has better individual accuracy and lower similarity between individuals. Meanwhile, the empirical performance of FWCRF is not inferior to the classical RF and AdaBoost on high-dimensional data. Furthermore, compared with other multivariate RFs, the advantage of FWCRF is that it can directly deal with the categorical features, instead of the conversion from the categorical features to the numerical features.

1. INTRODUCTION

The performance of classifiers can be significantly improved by aggregating the decisions of several classifiers instead of using only a single classifier. This is generally known as ensemble of classifiers, or multiple classifier systems. For an ensemble to achieve better accuracy than individual classifier, it is critical that there should be sufficient independence or dissimilarity between the individual classifiers.

The decision trees (DTs) are unstable classifiers, because when the training set is changed slightly, the prediction results may change greatly [1]. Therefore, DTs become the ideal candidates as the base classifiers in various ensemble learning frameworks. Random forest (RF) is a paragon of such ensembles, and is suitable for the classification and regression tasks of high-dimensional data and ill-posed data. RF is widely used in gene expression [2], credit risk prediction [3] and other fields [4] because of its excellent generalization ability, fast training speed, parallel processing, robustness to noise, few parameters, embedded feature selection.

RF is proposed by Breiman in [5]. In that paper, bootstrap sampling was used on training set to obtain a large number of random sampling subsets, and a DT was constructed based on each sampling subset. To further reduce the similarity between trees in the forest and speed up the construction of the trees, a subset of the features is selected randomly when each node in a tree is split, and the suitable split hyperplanes could be searched in the subspaces of features. In the prediction, the tree votes are combined by majority voting.

In contrast to the Adaboost [6], RF has obvious advantages in the construction speed and the robustness, which are attributed to the adoption of "random subspaces" strategy in node splits. As Ho said in [7]: "While most other classification methods suffer from the curse of dimensionality, this method can take advantage of high dimensionality." RF is very suitable for the classification and regression tasks of the high-dimensional and sparse data. However, for low-dimensional data, the DTs in RF have higher similarity which leads to the poor generalization ability of ensemble classifier. To improve this situation, the oblique DTs are considered as base classifiers. Murthy *et al.* [8] shows that there are $(n \cdot p)$ different axis-parallel split hyperplanes in the p-dimensional feature space with n samples. However, there are $2^p \cdot \binom{n}{p}$ different oblique split hyperplanes. Obviously, when constructing the DTs, more diverse base classifiers will be generated if the proper oblique splits are used at each internal node.

In Breiman [5] posed a simple random oblique split method. But according to the experimental results, we can see that there was no advantage compared with the axis-parallel split. With the emergence of oblique DT algorithms, some oblique split methods with low computational consumption are used in ensembles [9–11]. These ensembles inherit the characteristics of RF, such as high efficiency, adapting to high-dimensional data, and have exceeded the performance of classic RF in some application fields. However, the oblique split methods are only adapted to the numerical data. When

* *Corresponding author. Email: liuzhenyu@neusoft.edu.cn*

handling the data containing discrete and disordered categorical features, these ensembles need to convert the categorical features to numerical features by CRIMCOORD [12], OneHot encoder or other methods. The conversion may bring new bias to the classification tasks and reduce the generalization ability of classifiers. In Bjoern *et al.* [10], it has been shown that the performance of these methods on categorical data is not as good as that of classical RF.

In this paper, we present a method of node splits based on feature-weighting and clustering, which can not only combine multiple numerical features, but also combine multiple categorical features without conversion. And this split method working with the strategy of "random subspace" can construct more diverse DTs for ensemble. Therefore, based on the framework of RF, we use this split method to construct DTs. The ensemble of the DTs is called FWCRF. The experiments show that FWCRF has excellent generalization ability on numerical data, categorical data and mixed data. Especially, on the low-dimensional data, the performance of FWCRF outperforms classical RF.

The rest of the paper is as follows: Section 2 is the related work about brief reviews. The details of our split method and ensemble method will be given in Section 3. In Section 4, we compare the FWCRF with AdaBoost and RF through experiments. Section 5 gives our conclusion.

2. RELATED WORKS

In this section, we will briefly review node splits in DTs and ensembles of DTs first.

2.1. Split Methods

In Ho [7], on the premise of only considering the numerical data, the author divided the linear splits of nodes into three classes: axis-parallel linear splits, oblique linear splits and piecewise linear splits. The axis-parallel methods only act on one dimension at a time and thus result in an axis-parallel split. Besides, a suitable cut point (θ_l) needs to be searched. The samples in the node are divided into the left child node or the right child node with (1), according to their values (x_l) on the feature A_l.

$$x_l \leq \theta_l \tag{1}$$

The famous C4.5 [13] and CART [14] are axis-parallel trees, which have the advantages of fast induction and interpretability. However, if candidate features are strongly correlated, an extremely bad situation may occur. In Figure 1, it is easy to see that the features A_1 are strongly related to each other. To completely separate the two classes of samples, multiple axis-parallel splits are needed which will result in a complex DT structure. So, the axis-parallel splits reduce the interpretability of the model and lead to over-fitting due to sufficiently detailed splits, thus influencing the generalization ability.

The oblique split methods can deal with the problem shown in Figure 1. In the space of p-dimensional numerical features, the samples in the node are divided into the left node or right node according to (2).

$$\sum_{i=1}^{p} a_l x_l \leq \theta \tag{2}$$

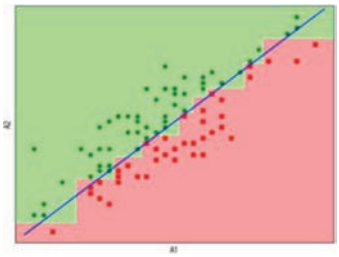

Figure 1 | Axis-parallel splits and oblique split.

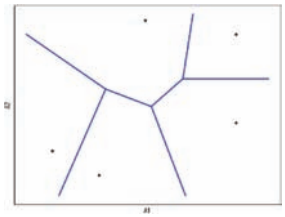

Figure 2 | Piecewise linear splits.

In (2), a_l is the coefficient of the lth feature, and if the $(p+1)$ coefficients including θ are determined, the split hyperplane can be uniquely determined. According to (2), the split line in Figure 1 is represented by the solid line, which can be described as follows: $A_1 - A_2 \leq 0$. Compared with the axis-parallel splits, the samples can be separated by only single split. Therefore, it is generally believed that the oblique splits can often produce smaller DTs and better generalization performance for the same data. In recent decades, a large number of oblique split methods have been proposed. CART-LC [14] is the oblique split method based on CART, and it used hill-climbing method to randomly disturb the coefficients of each feature in the space to determine the hyperplane. SADT [15] used simulated annealing algorithm to search hyperplanes to avoid falling into the local optimal solution. OC1 [8] combined CART-LC and SADT. Because thousands of candidate hyperplanes need to be tried, the time complexity of the above three algorithms is very high. FDT [16] is an oblique DT constructed by LDA. Restricted by the Fisher's discriminant analysis (FDA), FDT can only be applied to binary classification problems. In Hong *et al.* [17], the authors tried three methods (FDA, sparse LDA and ridge regression) to determine the hyperplanes. The HHCART [18] first obtained the eigenvectors of the original data, and then mapped the samples in training set to a new coordinate space, where the best axis-parallel hyperplane is searched (equivalent to the oblique hyperplane of the original space). A common drawback of these oblique split methods is that they cannot be directly applied to categorical data

Different from binary-way splits of axis-parallel and oblique methods, piecewise linear method is multi-way splits. As Figure 2 illustrated, piecewise linear splits are regarded as the combination of multiple oblique splits. The methods can be simply described as: looking for k anchors in feature space, and each sample is clustered according to the nearest neighbor anchor. The anchors here may be selected among training samples, class centroids or cluster centers. Pedrycz and Sosnowski [19], Li *et al.* [20] and our method fall into this class.

2.2. DT Ensembles

Boosting [6], Bagging [21], Random Subspaces [7] and RF [5] are classic in ensemble learning.

AdaBoost is the most famous representative of Boosting. In Boosting, the individual classifiers are added one at a time so that each subsequent classifier is trained on data which have been "hard" for the previous ensemble members. A set of weights is maintained across the samples in the data set so that samples that have been difficult to classify acquire more weight, forcing subsequent classifiers to focus on them. The weights are also used in final decision combination, and the weights are determined by accuracies of individual classifiers.

In Bagging, the training subsets for generating the individual classifiers are extracted from the original training set based on bootstrap method, and the majority voting method is used for decision of ensembles. Bagging is suitable for parallel implementation for fast learning. However, the diversity obtained by bootstrapping is not as good as Boosting.

Different from the Boosting and Bagging based on random subsampling, the random subspace method selects feature subsets randomly to construct individual classifiers. The experiments in [7] show that random subspace method can produce more diverse individuals than Boosting and Bagging.

RF is an extension of Bagging, which combines random subsampling and random subspace. RF not only inherits the parallelizable characteristic of Bagging, but also has fewer randomly selected features in the process of constructing the individual trees, which makes RF achieve the accuracy compared favorably with AdaBoost, and the construction speed is much faster than the latter.

The effectiveness of RF does not depend on the specific methods of DT induction, and the oblique splits and piecewise linear splits can also be used to construct trees in forest, such as principal components analysis (PCA) [9,22], linear discriminant analysis (LDA) [10], support vector machine (SVM) [11] and C-fuzzy clustering [23]. In some cases, these methods can bring about better individual "accuracy" and "diversity." However, PCA, LDA and SVM cannot be directly applied to categorical data.

3. FWCRF

The generalization ability of ensemble depends not only on the accuracy of individual DTs, but also on individual diversity. When the samples and features are sufficient, the univariate DTs with fast construction speed are more suitable to be the base classifiers of the ensembles. Instead, if the samples or features are insufficient, the multivariate split methods will bring about more diverse individuals for the ensembles. However, most multivariate split methods can only be used for numerical data. The feature-weighting and clustering split method proposed in this paper is based on the combination of multiple variables, and it is suitable to deal with the data of numerical features, categorical features and mixed features. The DTs constructed by this method have better generalization ability than the univariate DTs. The split method proposed can bring more diverse individuals to the ensembles by combining the "subsampling" and "subspace," so that it ensures the better generalization ability of the ensembles compared with the classical RF in case of fewer training samples or features.

3.1. Feature-Weighting and Clustering Split

The split method proposed is based on clustering assumption. The clustering assumption states that the samples belonging to the same cluster belong to the same class. K-means is a widely used clustering algorithm. In this paper, we adopt the improved K-means to split nodes.

The number of classes of the current node is regarded as the clustering number k. The n samples in training set D will be divided into k disjoint subsets, $C_1, C_2, \ldots, C_k$. Firstly, class centroids are treated as the initial centers of k clusters, respectively, represented by $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, ..., \boldsymbol{\mu}_{\mathrm{k}}$. Secondly, the $label_i$ is calculated by (3).

$$label_i = \underset{1\le j\le k}{\operatorname{argmin}} \|\boldsymbol{x_i} - \boldsymbol{\mu_j}\| \tag{3}$$

In (3), $\|\boldsymbol{x_i} - \boldsymbol{\mu_j}\|$ indicates the distance between $\boldsymbol{x_i}$ and μ_{j}. After each sample obtains the corresponding label, we use (4) to update the center of each cluster.

$$\boldsymbol{\mu_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} \boldsymbol{x_i} \quad \boldsymbol{x_i} \in C_j \tag{4}$$

Repeat (3) and (4) until the preset number of iterations is reached or the positions of all cluster centers are no longer changed.

The original K-means is an unsupervised clustering algorithm, which is suitable for unlabeled data. And the optimization goal is to minimize (5).

$$E = \sum_{j=1}^{k}\sum_{i=1}^{n_j} \|\boldsymbol{x_i} - \boldsymbol{\mu_j}\|^2 \boldsymbol{x_i} \in C_j \tag{5}$$

The goal of split is to reduce the class impurity of current node as much as possible. Note that the two goals are not the same. Therefore, we estimate the correlations between features and label to weight features. When calculating the distance from a sample to a cluster center, we give a larger weight to the feature strongly related to the label that enlarges the contribution of the feature to the distance. Otherwise, we give a smaller weight that reduces the contribution of the uncorrelated feature to the distance. In this way, the optimization goal of K-means algorithm is close to that of node split.

In this paper, we use the Relief-F algorithm [24,25] to weight features. The algorithm picks m samples. For each sample R, its k_{nn} nearest neighbors are searched in each class. The weight of feature A is calculated as follows:

$$\begin{aligned} W(A) = W(A) &- \sum_{j=1}^{k_{nn}} diff(A, R, H_j)/(mk_{nn}) \\ &+ \sum_{c\notin class(R)} \left[\frac{p(C)}{1-p(class(R))}\sum_{j=1}^{k_{nn}} diff(A, R, M_j(C))\right]/(mk_{nn}) \end{aligned} \tag{6}$$

where H_j indicates the jth nearest *neighbor* of R(H_j and R have the same label), $p(C)$ calculates the difference between two samples R_1 and R_2 on the feature A, as follows:

$$diff(A, R_1, R_2) = \begin{cases} \dfrac{|R_1[A] - R_2[A]|}{max(A) - min(A)} & if\ A\ is\ numerical \\ 0 & if\ A\ is\ categorical\ and\ R_1[A] = R_2[A] \\ 1 & if\ A\ is\ categorical\ and\ R_1[A] \neq R_2[A] \end{cases} \tag{7}$$

In order to illustrate the role of feature weighting in clustering, we use dataset *iris* to carry out a simple experiment: 150 samples of dataset *iris* come from three classes, and each class has 50 samples. We directly use K-means algorithm to cluster, and obtain 10 misclassified samples. The specific results are shown in the confusion matrix in Table 1.

Then, we use the Relief-F algorithm to calculate the weights of four features, which are 0.09, 0.14, 0.34 and 0.39 respectively. In the process of K-means clustering, the distances between samples and cluster centers are calculated by (8), where p indicates the number of features, w_l indicates *the weight of the lth feature.* We obtain 6 misclassified samples, and the specific results are shown in Table 2.

$$\|x_i - \mu_j\| = \sqrt{\sum_{l=1}^{p} w_l \cdot \left(x_{i,l} - \mu_{j,l}\right)^2} \tag{8}$$

Our method can be simply described as feature-weighting and clustering two steps. The specific process is shown in Algorithm 1.

In Algorithm 1, I_{max}value in the range of 1–10 with the probability of uniform distribution.

In step 3 of Algorithm 1, Relief-F is used to get the weights. Time complexity of Relief-F is $O\left(m \cdot p \cdot n \cdot \log_2 k_m\right)$, where p is the feature number, n is the instance number, m is the sampling number and k_{nn} is negligible, so the time complexity of Relief-F in this paper is $O\left(p \cdot n \cdot \log_2 n\right)$.

Steps 7 to 12 are the clustering process, and the time complexity is $O\left(I \cdot p \cdot n \cdot k\right)$, where k is cluster number and I is iteration number. When we use Algorithm 1 to split nodes, the max iterations I_{max} is no more than 10, it means that time complexity may reach $O\left(10 \cdot p \cdot n \cdot k\right)$ in the worst case.

Table 1 Split results of unweighted features for *iris.*

	setosa	versicolor	virginica
Child node 1	50	0	0
Child node 2	0	44	4
Child node 3	0	6	46

Table 2 Split results of weighted features for *iris.*

	setosa	versicolor	virginica
Child node 1	50	0	0
Child node 2	0	48	4
Child node 3	0	2	46

Algorithm 1: : split

Input: Current node training dataset D
Output: Divide D as $C_1, C_2, \ldots, C_k$, cluster centers $\mu_1, \mu_2, \ldots \mu_k$, feature subset F, the weight vector of features $\boldsymbol{w}$
1: Initilize the number of clusters k with the number of classes in D.
2: Select feature subset F, randomly.
3: Input D and F, call Relief-F to generate $\boldsymbol{w}$.
4: Assign w_{max} with maximum in $\boldsymbol{w}$, F excludes features whose weights are less than $\beta \cdot w_{\max}$, $\beta \in [0, 1]$, 0.2 by default.
5: Initialize $\mu_1, \mu_2, \ldots \mu_k$ by using the class centroids of samples in D.
6: **For** 1 to I_{max} **Do**
7: Combining (3) and (8), divide samples into $C_1, C_2, \ldots, C_k$.
8: Recalculate $\mu_1, \mu_2, \ldots \mu_k$ according to (4).
9: **If** $\mu_1, \mu_2, \ldots \mu_k$ do not change significantly **Then break For**
10: **End For**
11: **Return** $C_1, C_2, \ldots, C_k$, $\mu_1, \mu_2, \ldots \mu_k$and $\boldsymbol{w}$.

Considering the above two parts, the time complexity of the Algorithm 1 is $O\left(\left(10k + \log_2 n\right) \cdot p \cdot n\right)$. Compared with the time complexity of the classical axis-parallel splits, there's an extra k.

OC1 [8] is a classic oblique DT, whose time complexity is $O\left(p \cdot n^2 \cdot \log_2 n\right)$ in the worst case. In Wickramarachchi *et al.* [18], the time complexities of HHCART(A) and HHCART(D) are $O\left(\left(p + n \cdot \log_2 n\right) \cdot p^2 \cdot k\right)$ and $O\left(\left(p + \log_2 n\right) \cdot p \cdot n \cdot k\right)$ respectively. In Hong *et al.* [17], the speed of FDT for splitting node is close to or even better than that of axis-parallel split method. The time complexity of this method is $O\left(p^2 \cdot n\right)$. Unfortunately, it can only be applied in binary classification problems.

In summary, when k is small, the efficiency of the proposed split method is close to classical axis-parallel split method, and is better than most oblique split methods.

3.2. Categorical Feature

As mentioned in the previous subsection, the split method can be directly applied to numerical features. For categorical features, Relief-F algorithm can still be used to weight features. However, in the process of clustering, the representation of cluster center and the distance from sample to cluster center need to be redefined.

Inspired by the algorithms K-modes [26] and K-summary [27], in this paper, we use the probability estimation of each categorical feature value to represent the cluster center, and define a function to calculate the distance from sample to cluster center.

Definition 1. C_{j,x_l} represents the set of samples with value of x_l on the feature A_l in C_j, where $x_l \in \omega_1, \omega_2, \ldots, \omega_{d^{(l,j)}}$. The condition probability is estimated as follows:

$$P\left(x_l | j\right) = \frac{\left|C_{j,x_l}\right|}{\left|C_j\right|} \tag{9}$$

$S_{j,l}$ is the summary of all values of A_l in C_j, and defined as follows:

$$S_{j,l} = \left\{P\left(\omega_1 | j\right), P\left(\omega_2 | j\right), \cdots, P\left(\omega_{d^{(l,j)}} | j\right)\right\} \tag{10}$$

Definition 2. The center of C_j is represented by the following vector:

$$\boldsymbol{\mu}_j = \left(S_{j,1}, S_{j,2}, \cdots, S_{j,p}\right) \tag{11}$$

Definition 3. $diff\left(A_l, \omega, S_{j,l}\right)$ represents the distance between value ω and $S_{j,l}$ for A_l:

$$diff\left(A_l, \omega, S_{j,l}\right) = \begin{cases} 1 - P\left(\omega|j\right) & if\, \omega \in \left\{\omega_1, \omega_2, \cdots, \omega_{d(l,j)}\right\} \\ 1 & others \end{cases} \tag{12}$$

Definition 4. $dis\_c\left(\boldsymbol{x_i}, \boldsymbol{\mu_j}\right)$ represents the distance between sample x_i and center $\boldsymbol{\mu_j}$:

$$dis\_c\left(\boldsymbol{x_i}, \boldsymbol{\mu_j}\right) = \sum_{l=1}^{p} \left(w_l \cdot diff\left(A_l, x_{i,l}, S_{j,l}\right)\right) \tag{13}$$

Let's use an example to illustrate the above definition. Suppose there are two clusters C_1 and C_2 described by two categorical features A_1 and A_2, and each cluster contains 10 samples as is shown in Table 3.

According to (10) and (11), the centers of C_1 and C_2 are $\boldsymbol{\mu_1} = \left(S_{1,A_1}, S_{1,A_2}\right)$ and $\boldsymbol{\mu_2} = \left(S_{2,A_1}, S_{2,A_2}\right)$ respectively, where $S_{1,A_1} = \{\mathrm{a11} : 0.9, \mathrm{a12} : 0.1\}$, $S_{1,A_2} = \{\mathrm{a21} : 0.4, \mathrm{a22} : 0.4, \mathrm{a23} : 0.2\}$, $S_{2,A_1} = \{\mathrm{a11} : 0.4, \mathrm{a12} : 0.3, \mathrm{a13} : 0.3\}$, $S_{2,A_2} = \{\mathrm{a21} : 0.4, \mathrm{a22} : 0.4, \mathrm{a24} : 0.2\}$. Suppose that there is a sample $\boldsymbol{q} = (\mathrm{a11}, \mathrm{a23})$, the weights of A_1 and A_2 are $w_1 = 0.7$ and $w_2 = 0.1$ respectively. According to (12) and (13), the distances between sample $\boldsymbol{q}$ and two cluster centers ($\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$) are $0.15 = 0.7 * (1 - 0.9) + 0.1 * (1 - 0.2)$ and $0.52 = 0.7 * (1 - 0.4) + 0.1 * 1$, respectively. It means that $\boldsymbol{q}$ is closer to C_1.

Only considering the categorical feature data, in Algorithm 1, we use (11) to represent clustering center instead of the original ones in step 5 and 8, and use (13) to calculate the distance from sample to cluster center instead of (8) in step 7.

For mixed feature data, the vector of clustering center consists of two parts: one is the means of numerical features, and the other is the vector as shown in (11). In this case, we use (14) to calculate the distance from sample to cluster center, where *dis_n* and *dis_c* are obtained by (8) and (13) respectively. And γ is a real number between 0 and 1, which is used to adjust the proportion of two terms in the (14). In this paper, we use the method of uniform distribution to generate γ randomly, in order to generate more diverse individual classifiers.

$$dis\left(\boldsymbol{x_i}, \boldsymbol{\mu_j}\right) = (1 - \gamma) \cdot dis\_n\left(\boldsymbol{x_i}, \boldsymbol{\mu_j}\right) + \gamma \cdot dis\_c\left(\boldsymbol{x_i}, \boldsymbol{\mu_j}\right) \tag{14}$$

Table 3 The distribution of values.

	A_1	A_2
C_1	a11:9, a12:1	a21:4, a22:4, a23:2
C_2	a11:4, a12:3, a13:3	a21:4, a22:4, a24:2

3.3. The Constructions of DTs

In FWCRF, the trees are constructed on the basis of the top-down and recursion, and the split should be stopped, only when all samples in the current node are the same class or the feature subset used cannot distinguish these samples. The specific process is shown in Algorithm 2.

Algorithm 2: : create_tree

Input: training set D_s
Output: the decision tree
1: Create *node* according to the samples in D_s.
2: **Procedure** *grow*(*node*)
3: **If** All the samples are of the same class or have the same feature values **Then**
4: Mark the *node* as a leaf, and label it with the class of the majority of samples in D_s.
5: **Return** *node*
6: **End If**
7: Call *split* to get the cluster $C_1, C_2, \ldots, C_k$, $\mu_1, \mu_2, \ldots \mu_k$, F and $\boldsymbol{w}$.
8: Save the values of $\mu_1, \mu_2, \ldots \mu_k$, F and $\boldsymbol{w}$ into the current *node* for the prediction.
9: **For** i = 1 to k **Do**
10: Create *node_i* according to samples in C_i, call *grow*(*node_i*).
11: **End For**
12: **End Procedure**
13: **Return** the *node*-rooted decision tree.

3.4. The Forest Construction and Combination Prediction

RF and Bagging use bootstrap method to independently and randomly select the subsets of the raw training samples with replacement. In this way, some samples will appear repeatedly in the same sample subset. However, we don't follow the method when we build the forests. The reason is that we use the Relief-F algorithm to weight the features in the split processing. The duplicate samples will affect the selection of the neighbors and cause feature weight to be enlarged. FWCRF extracts n_s samples from the original training set D to form a subset D_s, using a randomly stratified sampling method without replacement. The n_s is a parameter determined by users. The D_s is used to train an individual classifier. The data that are not sampled consists of out-of-bag dataset D_o, $D_s \cup D_o = D$ and $D_s \cap D_o = \emptyset$. According to (15), we use the samples in D_o to estimate the voting confidence degree α of each leaf node in the corresponding tree.

$$\alpha = (acc + 1) / (acc + err + 2) \tag{15}$$

In (15), the *acc* and *err* represent the number of correctly classified samples and misclassified samples of D_o on the leaf node, respectively.

n_t and n_f are the parameters in FWCRF, and n_t represents the size of forest and n_f determines the dimension of random subspace F in Algorithm 1. Users can adjust these parameters to trade off the accuracy and diversity of individual classifiers to achieve satisfactory ensemble accuracy and execution speed. The reason why these parameters should be decided by the users is that the way that can

be applied to all classification problems is not available at present. The forest construction process is shown in Algorithm 3.

Algorithm 3: : create_forest

Input: training set D, n_s, n_t and n_f
Output: the forest
1: Create an empty forest Fo.
2: **For** i = 1 to n_t **Do**
3: Sample randomly to obtain D_s and D_o via D.
4: Use D_s to generate T_i.
5: Use D_o to calculate the voting confidence of each leaf node in T_i.
6: Add T_i to F_o.
7: **End For**
8: **Return** F_o

The combination prediction is realized by (16).

$$Fo(\mathrm{x}) = \arg\max_{1\le j\le k} \sum_{i=1}^{n_t} \alpha_i \cdot I\left(T_i(\mathrm{x}) = j\right) \tag{16}$$

In (16), $Fo(\boldsymbol{x})$ and $T_i(\boldsymbol{x})$ represent the decisions of the forest and the ith tree on the sample $\boldsymbol{x}$, and α_i represents the voting confidence degree of the leaf node which the sample $\boldsymbol{x}$ fell into, and I(·) is the indicator function. Compared with the majority voting, our combination prediction function only uses the confidence degree of the corresponding leaf node instead of the constant 1.

4. EXPERIMENT

In this section, we implement experiments to compare the performance of FWCRF with classic RF and AdaBoost. Firstly, we introduce the datasets and the main parameters of the three ensembles in the experiments. Then, we analyze the classification results. Finally, we choose two datasets to compare the anti-noise ability of three classifiers.

4.1. Experiment Set

In UCI [28], we select a benchmark of 31 datasets for experiments. From the perspective of features, these datasets include numeric, categorical and mixed data. The number of prediction features ranges from 3 to 240. The number of the samples ranges from 101 to 20000, and the number of class labels ranges from 2 to 26. All data have no missing values. As shown in Table 4, Rows 1–21 are the numerical datasets and Rows 22–26 are the categorical datasets and the others are mixed datasets.

The original data set *abalone* in Table 4 is a 29-classification problem. However, in our experiments, it will be regarded as a 3-classification problem, with 1–8 as a group, 9–10 as a group and the rest as a group.

We use C++ language to realize what we propose. In each split, we use Relief-F for feature-weighting, the number of nearest neighbors k_{nn} values 1 and the number of sampling m is set $\lceil \log_2 n_{cur} \rceil$, where n_{cur} is the number of the samples in current node. When clustering, I_{max} is uniformly distributed random number ($I_{\max} \in \{1, 2, ..., 10\}$), and γ is uniformly distributed random decimal ($\gamma \in [0, 1]$). In the ensemble, n_t is 100, and n_s is equal to $\lceil 0.7n \rceil$ n_f is obtained by $\lceil \log_2 p \rceil$ (p is the number of features).

Table 4 Datasets.

No.	Name	Samples	Features	Classes
1	Blood	748	4	2
2	BC	569	30	2
3	Ecoli	336	7	8
4	EEG	14980	14	2
5	Glass	214	9	6
6	HS	306	3	2
7	Image	2130	19	7
8	Iris	150	4	3
9	Letter	20000	16	26
10	MF-fac	2000	216	10
11	MF-fou	2000	76	10
12	MF-kar	2000	64	10
13	MF-mor	2000	6	10
14	MF-pix	2000	240	10
15	MF-zer	2000	47	10
16	Sonar	208	60	2
17	Vehicle	846	18	4
18	Wav1	5000	21	3
19	Wav2	5000	40	3
20	Wine	178	13	3
21	Yeast	1484	8	10
22	Balance	625	4	3
23	Car	1728	6	4
24	Chess	3196	36	2
25	Hayes	160	4	3
26	MONK	432	6	2
27	Abalone	4177	7&1	3
28	CMC	1473	2&7	3
29	Flags	194	10&19	8
30	TAE	151	1&4	3
31	Zoo	101	15&1	7

We choose the widely used Adaboost and RF methods as the baseline in this paper. The two algorithms use the corresponding components of Weka [29]. Among them, the base classifier of AdaBoost is set to j48 (C4.5 implemented in Weka), the ensemble size is set to 100 and other settings adopt default values. Because of the fast generation of RF, the number of the individual classifiers of RF is set to 500, and the other settings adopt the default values.

We refer to [30]. The 10 repetition of 10-fold cross validation is used in our experiments to report the average accuracies of three ensembles on the test sets. Friedman test and Nemenyi test will be used to validate the differences of algorithms.

4.2. The Results and Analysis

Table 5 lists the accuracies obtained by the three ensembles on 31 datasets. The last row of the table is the average result. The FWCRF achieves the best accuracy on 20 of the datasets. The average accuracy of FWCRF is 84.92% on 31 datasets, which slightly outperforms AdaBoost (83.40%) and RF (83.59%). In order to further demonstrate the differences of the ensembles, the Friedman test is used. We use the averages of the ranks of three ensembles on 31 datasets to calculate $F_F = 6.40152$. Here, F_F is $F(2, 60)$, and the critical value is $F_{2,60} = 3.15$. So, we reject the null hypothesis. And then we calculate $CD = 0.595377$ based on Nemenyi test. In this case of these conclusions, the FWCRF has obvious performance advantages compared with AdaBoost and RF as we expected.

In order to demonstrate the advantages of FWCRF in categorical data and low-dimensional data, we first divide 31 datasets into three groups according to the types of features. On 5 categorical datasets, the average accuracies of FWCRF, AdaBoost and RF are 93.51%, 89.60% and 90.64% respectively, and the average accuracy of FWCRF is about 3% higher than that of the other two classifiers. On 5 mixed datasets, the average accuracies of FWCRF, AdaBoost and RF are 69.73%, 65.23% and 68.30% respectively, and FWCRF and RF are significantly better than AdaBoost. On 21 numerical datasets, the three classifiers perform equally, and their average accuracies are 86.51%, 86.24% and 85.55%, respectively.

Then we group the 31 datasets by the number of features. The datasets with less than 10 features is one group and the others are another group. On 14 low-dimensional datasets, the average accuracies of FWCRF, AdaBoost and RF are 78.37%, 75.14% and 75.62%, respectively, and the average accuracy of FWCRF is about 3% higher than that of the other two classifiers. On another group, the three classifiers perform equally, and their average accuracies are 90.33%, 90.20% and 90.15%, respectively.

For the convenience of analysis, we sort the datasets by the D-values of the accuracies of FWCRF and RF, and fill them in Table 5. And a result is found that besides *Sonar* in the first 5 rows, the other four datasets only include 3 or 4 features respectively. It is demonstrated that the proposed multivariate split method can generate more diverse individual classifiers than the univariate split method on the low-dimensional data, thus improving the classification accuracy of the ensemble. To further verify the result, we follow the method in [7] to calculate the accuracies of individual classifiers in the ensemble and the similarities between them. The individual accuracy is the average of the accuracies obtained by all individuals in the ensemble on the test sets. And the similarity is averaged over all 4950 pairs of 100 trees (124750 pairs in RF). The similarity ($S_{i,j}$) of the pair of individuals is calculated as (17).

$$S_{i,j} = \frac{1}{n}\sum_{k=1}^{n} \mathrm{I}\left(T_i(\mathrm{x_k}) = T_j(\mathrm{x_k})\right) \tag{17}$$

where, n is the number of test samples, $T_i(\boldsymbol{x_k})$ and $T_j(\boldsymbol{x_k})$represent the decisions of individual i and j on test sample $\boldsymbol{x_k}$, respectively.

We choose *HS, Balance, Sonar, Letter* and *EEG* for similarity comparison experiment, and the results are reported in Table 6. The performance of FWCRF is significantly better than RF on low-dimensional data *HS* of numerical feature, low-dimensional data *Balance* of categorical feature and high-dimensional data *Sonar*. FWCRF and RF have the similar performance on *Letter*. However, the performance of FWCRF is inferior to RF on *EEG*.

The *HS* only contains three numerical features, and the accuracies of the three ensembles are not improved obviously compared with the corresponding individuals, and FWCRF obtains the best ensemble accuracy due to the highest individual accuracy and the lowest similarity. On *Balance* with four categorical features, although RF obtains the highest individual accuracy, FWCRF has the highest ensemble accuracy due to the lowest similarity between individuals. AdaBoost's lowest accuracy on *Balance* is attributable to its poor individual accuracies. On the *Sonar*, the ensemble accuracy of FWCRF is approximately 4% higher than the other two. The reason why FWCRF can beat AdaBoost is its lower individual similarity, and the reason why FWCRF can beat RF is its better individual accuracy. On the *Letter*, the three ensembles showed similar performances. On the *EEG*, the ensemble accuracy and individual accuracy of FWCRF are approximately 5% and 10% lower than the other two, respectively. By analyzing *EEG*, it is found that 7 of the 14 features have outliers. For example, the minimum value of certain feature is 1366.15, its maximum value is 715897, the mean is 4416.44, the median is 4354.87 and the maximum value is 162 times of the mean. The existence of outliers makes the obtained cluster centers seriously deviate from the ideal ones. Although this is conducive to the diversity of individuals, it decreases individual accuracy severely.

Considering the performance of the three ensembles on these five datasets with different dimensions and types, it is not difficult to find that under the univariate split methods, AdaBoost is more

Table 5 The accuracy of FWCRF compared with Adaboost and random forest (RF).

Dataset	Accuracy% (Rank)		
	Adaboost	RF	FWCRF
HS	70.82 (2)	65.95 (3)	73.53 (1)
Balance	75.15 (3)	79.54 (2)	86.45 (1)
Blood	77.66 (2)	72.83 (3)	78.14 (1)
Hayes	76.81 (3)	79.56 (2)	84.31 (1)
Sonar	85.17 (3)	85.58 (2)	89.76 (1)
CMC	50.46 (3)	50.49 (2)	53.76 (1)
Car	96.87 (2)	94.83 (3)	97.36 (1)
TAE	54.44 (3)	62.45 (2)	64.83 (1)
Ecoli	84.35 (3)	84.88 (2)	87.08 (1)
MF-mor	71.75 (2)	69.91 (3)	71.94 (1)
MF-zer	78.97 (2)	78.06 (3)	79.74 (1)
Iris	94.07 (3)	94.53 (2)	95.80 (1)
Abalone	62.80 (3)	63.88 (2)	64.66 (1)
BC	97.19 (2)	96.59 (3)	97.29 (1)
Wav1	85.69 (2)	85.49 (3)	86.13 (1)
Flags	62.11 (3)	68.81 (2)	69.33 (1)
Glass	78.27 (3)	78.50 (2)	79.02 (1)
Wav2	85.07 (3)	85.40 (2)	85.84 (1)
Letter	97.21 (1)	96.74 (3)	97.17 (2)
MF-kar	96.40 (3)	96.73 (2)	97.12 (1)
Zoo	96.34 (1)	95.84 (3)	96.04 (2)
Wine	97.30 (3)	97.64 (2)	97.81 (1)
Chess	99.66 (1)	99.30 (3)	99.41 (2)
MONK	99.49 (3)	99.98 (2)	100.00 (1)
Image	98.64 (1)	98.07 (2)	98.05 (3)
MF-fac	97.73 (1)	96.97 (2)	96.92 (3)
MF-pix	97.73 (2)	97.76 (1)	97.36 (3)
MF-fou	84.06 (2)	84.38 (1)	83.64 (3)
Yeast	58.99 (3)	61.34 (1)	60.24 (2)
Vehicle	78.16 (1)	75.08 (2)	73.96 (3)
EEG	95.91 (1)	94.21 (2)	89.77 (3)
average	83.40 (2.26)	83.59 (2.23)	84.92 (1.52)

Table 6 The Individual accuracy and similarity of FWCRF compared with Adaboost and rnadom forest (RF).

Dataset	Ensemble Accuracy% (Individual Accuracy %, Similarity%)		
	Adaboost	RF	FWCRF
HS	70.82 (69.82, 87.68)	65.95 (64.74, 89.76)	73.53 (71.31, 84.72)
Balance	75.15 (64.91, 67.37)	79.54 (76.91, 83.71)	86.45 (68.01, 63.30)
Sonar	85.17 (73.41, 78.28)	85.58 (71.25, 71.05)	89.76 (74.87, 68.04)
Letter	97.21 (88.09, 79.08)	96.74 (85.19, 74.62)	97.17 (84.16, 73.63)
EEG	95.91 (84.76, 75.05)	94.21 (84.14, 72.13)	89.77 (75.36, 68.84)

Note: HS, Haberman's Survival; EEG, EEG Eye State.

competitive than RF in low-dimensional datasets, which is mainly due to the lower similarity between individual classifiers brought by Boosting method compared with Bagging + subspace method. FWCRF also uses Bagging + subspace method, but FWCRF makes up for the deficiency of the classical axis-parallel split on low-dimensional datasets by utilizing the multivariate split method.

4.3. Robustness

For the classification problem, the noise existing in the prediction features of training data is called input noise, and the noise existing on the class label is called output noise. In Breiman [5], the author points out that RF is more robust than AdaBoost on data with output noise. FWCRF not only uses the classic framework of RF, but also combines the multiple features at each split. Therefore, we infer that FWCRF should has better anti-noise ability than the classic RF and AdaBoost. In this paper, we implement the comparison of robustness on the datasets of categorical feature *Car* and numerical feature *MF-mor*.

In the experiments of output noise, we randomly add m% noise to samples in training set, $m \in \{5, 10, 15, 20, 25, 30\}$, by changing the class label to another value different from the original value in the training set. The experiment results are shown in Figure 3. As expected, AdaBoost has the worst anti-noise ability on two datasets. When the noise ratio reaches 30%, the accuracies of AdaBoost are 18.52% and 14.32% lower than the results in Table 5. On *Car*, the anti-noise abilities of RF and FWCRF are similar, with the decreases of 8.34% and 8.53%, respectively. On *MF-mor*, the accuracy of RF decreases by 6.98%, while FWCRF decreases by 1.71% and has little interference from the noise.

In the experiments of input noise, we add m% noise to the values of the prediction features, $m \in \{5, 10, 15, 20, 25, 30\}$. On *Car*, we change the selected feature value to another one which is different from the original value and belongs to the domain of the feature. On *MF-mor*, we change the selected value of feature to a random number which follows the uniform distribution within the range of the feature value on the current training set. The experimental results are shown in Figure 4. The three ensembles show similar anti-noise capability. On *Car*, the accuracies of the three classifiers decrease monotonically with the increase of m, and the rate of the decline decreases gradually. Meanwhile, the three ensembles are almost unaffected by noise on *MF-mor*.

5. CONCLUSION

The RF [5] is known as a representative work of ensemble learning, which is suitable for high-dimensional data. However, it cannot get a good ensemble effect on low-dimensional data because the univariate split method cannot generate diverse individuals. Although the emergence of some oblique RFs based on numerical features partially solves the above problems, they cannot be directly applied to the datasets containing categorical features.

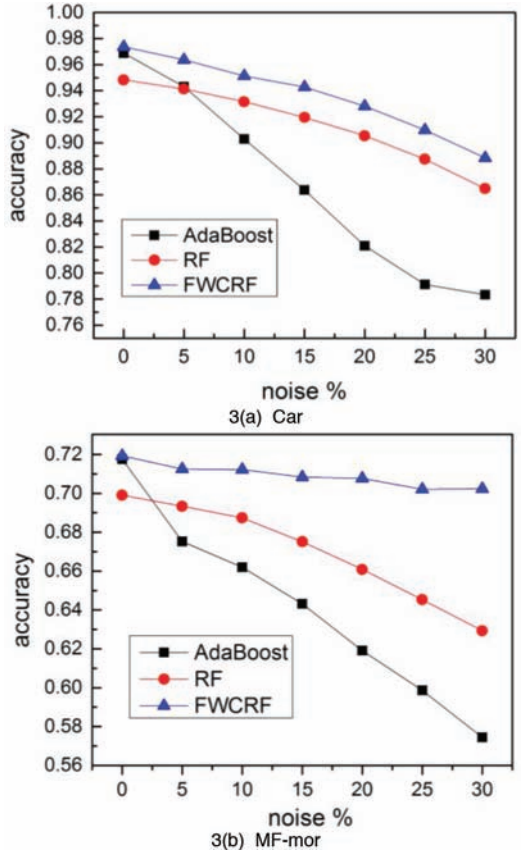

Figure 3 The Robustness of FWCRF compared with AdaBoost and random forest (RF) to output noise.

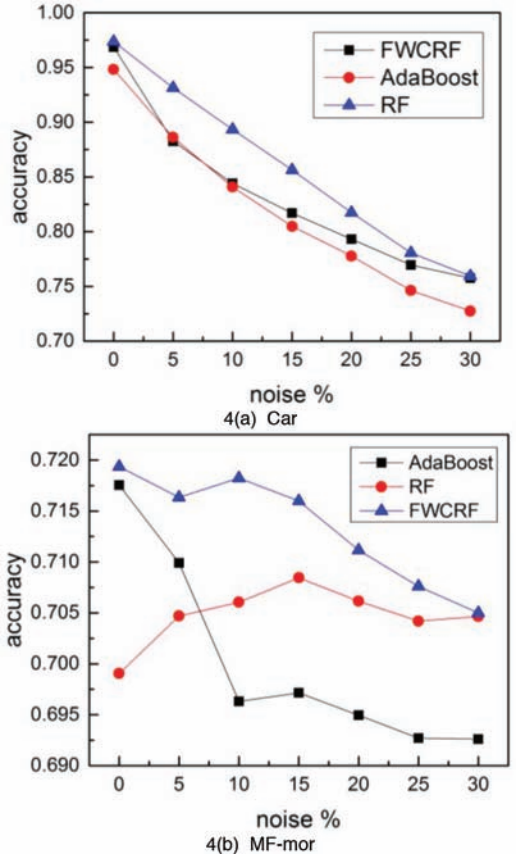

Figure 4 The Robustness of FWCRF compared with AdaBoost and random forest (RF) to iutput noise.

In this paper, we propose a feature-weighting and clustering method combining multiple variables. When splitting nodes, we first use Relief-F algorithm to weight features, and then use clustering algorithm based on weighted distance to split nodes. This method can be regarded as a variation of classical K-means. By defining the "center" and "distance between two points," we make this method applicable to the categorical features. We use the DTs generated by this method as the base classifiers, and construct the ensemble classifier (FWCRF) based on the framework of RF. Finally, we compare FWCRF with the classical RF and AdaBoost on 31 datasets. The experimental results show that the FWCRF outperforms the other two ensembles on low-dimensional datasets, whose performance is comparable with the other two on the rest datasets. At the same time, we also verify that the robustness of FWCRF is superior to AdaBoost on the data with output noise.

CONFLICT OF INTEREST

The authors declare that they have no competing interests.

AUTHORS' CONTRIBUTIONS

The study was conceived and designed by Zhenyu Liu and Tao Wen and experiments performed by Wei Sun and Qilong Zhang. All authors read and approved the manuscript.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous referees for their valuable comments and suggestions, which improved the technical content and the presentation of the paper. This work is supported by the National Nature Science Foundation of China under Grant 61772101, 61602075 and Key Natural Guidance Plan of Liaoning Province in 2018.

REFERENCES

[1] L. Breiman, Bias, Variance, and Arcing Classifiers, Technical Report 460, Statistics Department, University of California, Berkeley, CA, USA, 1996.
[2] S. Kimura, M. Tokuhisa, M. Okada, Inference of genetic networks using random forests: assigning different weights for gene expression data, J. Bioinf. Comput. Biol. 17 (2019), 1950015.
[3] A. Namvar, M. Siami, F.A. Rabhi, *et al.*, Credit risk prediction in an imbalanced social lending environment, Int. J. Comput. Intell. Syst. 11 (2018), 925–935.
[4] A.S. More, D.P. Rana, I. Agarwal, *et al.*, Random forest classifier approach for imbalanced big data classification for smart city application domains, Int. J. Comput. Intell. Syst. 1 (2020), 260–266.
[5] L. Breiman, Random forests, Mach. Learn. 45 (2001), 5–32.
[6] Y. Freund, R. Schapire, A decision-theoretic generalization of online learning and an application to boosting, J. Comput. Syst. 55 (1997), 119–139.
[7] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998), 832–844.
[8] S.K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, J. Artif. Intell. Res. 2 (1996), 1–32.
[9] J.J. Rodríguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: a new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006), 1619–1630.
[10] H. Bjoern, B. Menze, M. Kelm, D.N. Splitthoff, On oblique random forests,in Machine Learning and Knowledge Discovery in Databases European Conference (ECML PKDD 2011), Proceedings, Part II, Athens, Greece, 2011.
[11] R. Katuwal, P.N. Suganthan, L. Zhang, An ensemble of decision trees with random vector functional link networks for multi-class classification, Appl. Soft Comput. 70 (2018), 1146–1153.
[12] W.Y. Loh, Y.S. Shih, Split selection methods for classification trees, Stat. Sinica. 7 (1999), 815–840.
[13] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1992.
[14] W.L. Buntine, Learning classification trees, Stat. Comput. 2 (1992), 63–73.
[15] R.S. Bucy, R.S. Diesposti, Decision tree design by simulated annealing, ESAIM Math. Model. Num. Anal. 27 (1993), 515–534.
[16] A. López-Chau, J. Cervantes, L. López-García, *et al.*, Fisher's decision tree, Expert Syst. Appl. 40 (2013), 6283–6291.
[17] K.S. Hong, P.L. Ooi, C.K. Ye, *et al.*, Multivariate alternating decision trees, Pattern Recognit. 50 (2016), 195–209.
[18] D.C. Wickramarachchi, B.L. Robertson, M. Reale, *et al.*, HHCART: an oblique decision tree, Comput. Stat. Data Anal. 96 (2015), 12–23.
[19] W. Pedrycz, Z.A. Sosnowski, C-fuzzy decision trees, IEEE Trans. Syst. Man Cybern. Part C. 35 (2005), 498–511.
[20] Y. Li, E. Hung, K. Chung, *et al.*, Using a variable weighting k-means method to build a decision cluster classification model, Int. J. Pattern Recognit. Artif. Intell. 26 (2012), 1250003.
[21] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996), 123–140.
[22] H. Lu, Y. Meng, K. Yan, *et al.*, Kernel principal component analysis combining rotation forest method for linearly inseparable data, Cogn. Syst. Res. 53 (2019), 111–122.
[23] Ł. Gadomer, Z.A. Sosnowski, Knowledge aggregation in decision-making process with C-fuzzy random forest using OWA operators, Soft Comput. 23 (2019), 3741–3755.
[24] K. Kira, L.A. Rendell, A practical approach to feature selection, in Proceedings of the Ninth International Workshop on Machine Learning (ML 1992), Aberdeen, Scotland, 1992.
[25] I. Kononenko, Estimating attributes: analysis and extension of relief, in Proceeding of 1994 European Conference on Machine Learning, Catania, Italy, 1994, pp. 171–182.
[26] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Mining Knowl. Discov. 2 (1998), 283–304.
[27] S.Y. Jiang, Q.H. Li, An enhanced k-means clustering algorithm, Comput. Sci. Eng. 28 (2006), 56–59.
[28] M. Lichman, UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml
[29] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (2009), 10–18.
[30] R. Rivera-Lopez, J. Canul-Reich, Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach, IEEE Access. 6 (2018), 5548–5563.