Accepted Manuscript

Ensemble decision forest of RBF networks via hybrid feature clustering approach for high-dimensional data classification

Shadi Abpeykar, Mehdi Ghatee, Hadi Zare

PII: S0167-9473(18)30198-1
DOI: https://doi.org/10.1016/j.csda.2018.08.015
Reference: COMSTA 6676

To appear in: *Computational Statistics and Data Analysis*

Received date : 7 September 2017
Revised date : 15 August 2018
Accepted date : 19 August 2018

Please cite this article as: Abpeykar S., Ghatee M., Zare H., Ensemble decision forest of RBF networks via hybrid feature clustering approach for high-dimensional data classification. *Computational Statistics and Data Analysis* (2018), https://doi.org/10.1016/j.csda.2018.08.015

Ensemble decision forest of RBF networks via hybrid feature clustering approach for high-dimensional data classification*

Shadi Abpeykar [a], Mehdi Ghatee [a,†], Hadi Zare[b]

[a] Department of Computer Science, Amirkabir University of Technology, Tehran, Iran

[b] Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran

Abstract

Classification of the high-dimensional data is challenging due to the curse of dimensionality, heavy computational burden and decreasing precision of algorithms. In order to mitigate these effects, feature selection approaches that can determine an efficient subset of features are utilized in the processing. However, most of these techniques attain just one subset of non-redundant features including the best ones. Alternatively, clustering approaches can be used to find the most informative clusters of features instead of generating just a single subset. So called, Hybrid Feature Clustering (HFC) method is capable of maximizing the classification accuracy while keeping the amount of redundant features in each cluster low. The patterns of each cluster are classified by a neural tree that employs Radial Basis Function (RBF) for the nodes. Within each neural tree, a hierarchical approach is proposed to transfer the knowledge of synaptic weights from a parent RBF node to each child. A gating network is applied on the forest of these neural trees in order to aggregate the results. By assessing the classification accuracy and the computational complexity on high-dimensional datasets it can be shown that the proposed solution has outperformed the state of the art classifiers. Furthermore, the computational complexity and the convergence of this method are theoretically proven and the robustness analysis under noisy conditions are conducted.

Keywords: Big-Data with High-dimensional Features; Feature Clustering; Neural Tree; Knowledge Transferring; Gating Network; Ensemble Learning.

1 Introduction

Big data classification is important in wide range of scientific and industrial processes (Mahani & Sharabiani, 2015). The different challenges on big data make the traditional classifiers weak in dealing with these datasets (Ward & Barker, 2013). The most important challenges of big dataset

* The codes of this paper have been uploaded in the supplementary material section.

† The corresponding author: M. Ghatee is with Department of Computer Science, Amirkabir University of Technology, Hafez Ave., Tehran 15875-4413, Iran.
Fax: +9821 66497930, Tel: +98 21 64542531, Email Address: ghatee@aut.ac.ir, URL: www.aut.ac.ir/ghatee

are Velocity, Volume and Variety, which is called 3-Vs in researches (Douglas, 2001). In addition, these challenges were expanded to 4-Vs and 5-Vs by adding Veracity (Beyer & Laney, 2012) and Value (Anuradha, 2015). Recently, multitudes of works considered the challenges of big data in each of the aforementioned aspects (Napoli, et al., 2016), (Savvas, et al., 2013), (Shim, 2012). When the volume of data or number of features is very high, we face with the curse of dimensionality problem (Li & Liu, 2017). Now, we deal with this problem by considering datasets with feature dimension ranging from average size to more than 3 million features. Our approach is applying feature clustering method. Feature selection methods, including filter, wrapper, and hybrid methods try to choose the most related features to the class labels and to eliminate the redundant features, see e.g., (Dernoncourt, et al., 2014), and (Reynès, et al., 2008), which helps to deal with high-dimensional features (Bak & Jensen, 2016). The filters are independent from the classifiers and the learning stage. They are based on statistical and information theoretic criteria to select more relevant and less redundant features (Sheikhpour, et al., 2017). The wrapper methods select suitable subset of features based on the attained accuracies via the benchmark classification algorithms (Liu & Yu, 2005). Some researchers proposed some hybrid techniques to exploit the properties of wrapper and filter approaches simultaneously (Eroglu & Kilic, 2017). Generally, the filter techniques have more speed but less accuracy than the wrapper ones. Since both of training accuracy and redundancy should be improved, a hybrid method is proposed in this paper. Furthermore, in some of the traditional methods, the irrelevant features are removed but redundant features are not considered, which are failed in handling features structures (Frénay, et al., 2014). However, handling the feature structure and making feature clusters can improve the classification performance (Xu & Lee, 2015), (Song, et al., 2013), (Gayathri, 2014), (Gupta, et al., 2014), (Krier, et al., 2007), (García-Torres, et al., 2016) and (Goswami, et al., 2017). Therefore, this paper focuses on feature clustering rather than feature selection.

On the other hand, there are two main strategies to upgrade the classification algorithms for big datasets. The first is based on the software-hardware developments and the second is to develop fast and accurate learning algorithms. In the former strategy, one can exploit the strong capability of hardware and software platforms to accelerate classification process on big datasets. As some instances, different platforms like GPU architecture in pattern identification (Napoli, et al., 2016), distributed systems like cloud computing for big data classification (Hashem, et al., 2015), MapReduce platform for big data analysis (Shim, 2012), (Savvas, et al., 2013), (Triguero, et al., 2015) and Hadoop to achieve stronger and more rapid classifiers (Anuradha, 2015), (Savvas, et al., 2013) have been utilized. However, the latter strategy covers some developments of learning algorithms to accelerate the classification time and to achieve more accurate results. (Chen & Lin, 2014), (Savvas & Sofianidou, 2014) and (Savvas & Sofianidou, 2016) redesigned algorithms to deal with big datasets. We also focus on extending an ensemble learning algorithm (Xue, et al., 2016) for high-dimensional datasets. The contribution of ensembles is combining some base classifiers and feature selection methods to reduce the size of the feature space and to construct more powerful classifiers (Song, et al., 2016).

In this paper by noting to the group structure of features, we introduce a new feature clustering approach as the preprocess for an ensemble method. Similar to (Yijing, et al., 2016), we use neural tree with decision tree structure that includes neural networks like RBF networks in the nodes. (Yijing, et al., 2016) and (Tao, et al., 2013) proposed a unique classifier for all clusters that is a

drawback. In this paper, we apply a forest of multiple neural trees on the different clusters. Each neural tree uses the benefits of rule generation of decision trees and the generalization ability of RBF networks.

The rest of this paper is organized as the following. Section 2 introduces some fundamental concepts. Section 3 discusses the proposed feature clustering method. Section 4 describes the components of our proposed classifier. The performance analysis is presented in Section 5 . Section 6 is devoted to the efficiency analysis on some benchmark datasets. Finally, Section 7 concludes the paper with discussions.

2 Fundamental concepts

Feature selection methods, are widely used to decrease the dimensionality of features space of datasets. These methods usually lead to a single subset of dependent features to target classes, with the least inner redundancy. To achieve these non-redundant features, some of the high relevant features may be lost and the resulted subset of features is not robust. However, HFC clusters the features such that the features with high dependencies on class labels are remained. This method, firstly, eliminates the features whose mutual information is less than a pre-defined threshold. Then it uses a modified GA to cluster these features into several groups with the least inner redundancy. GA tries to maximize the classification learning accuracy and to minimize the inner redundancy between features. In the case of missing the effect of a feature in a cluster, the other clusters still have opportunities to consider this feature. Besides, all of the high-related features to the target classes are reachable in the clusters.

To classify the patterns based on the features of each cluster, we use a neural tree of RBF networks (Foresti & Pieroni, 1998), (Micheloni, et al., 2012), (Rani, et al., 2015), and (Abpeykar & Ghatee , 2018). By applying this classifier in each cluster, a Forest of Decision trees with RBF networks and Knowledge transferring (FDRK) is developed. The results of all neural trees of this forest are aggregated by a gating method. Since, the majority voting as a popular aggregation method cannot lead to a unique label in multiclass problems, a novel Mix gating network based on Majority vote and Cheat method (MMCF) has been proposed for FDRK. This gating network like a gardener that collects the best fruits from the trees, aggregates the best results of the neural trees and makes the final decision about the class labels. Since, any neural tree achieves a leaf node in each level, FDRK performs very well on classification problems.

Figure 1, presents the structure of FDRK and every neural tree of FDRK is shown in Figure 2. In each node of this tree, based on the RBF training performance, it can decide on data splitting for the next levels of the neural tree. The left child node contains samples of a single class that are selected correctly with the least training error. Other samples will be sent to the right child node. Gradually LTS (Local Training Set) becomes more restrictive until classifying all of the samples. The details will be described in the next subsection. To understand the process of FDRK, we give a simple example in Figure 3. This example assumes a dataset with 10 training samples, 8 features, and 2 classes. After removing the irrelevant features, five features remain in LTS. Assume that HFC defines two clusters of features. Two neural trees are built on these clusters of features. The RBF network in each node, classifies samples of LTS based on the corresponding features of cluster. Then, the training error for each class is calculated and the samples of LTS are split to two subsets; samples of a class with least training error which are correctly classified and the other

samples. The first subset is assigned to the left child and the second subset is assigned to the right child node and the classification process continues on the samples of the latter node. This process terminates when all of the samples find their labels. This kind of partitioning improves the training of the neural tree because all of the neural trees get leaves in their most left branches. In addition, it simplifies the computations of RBF networks because of decreasing the number of training samples in each node. Furthermore, for a new sample with a set of features similar to the features of cluster 1 and cluster 2, MMCF gating network assigns an appropriate class label.

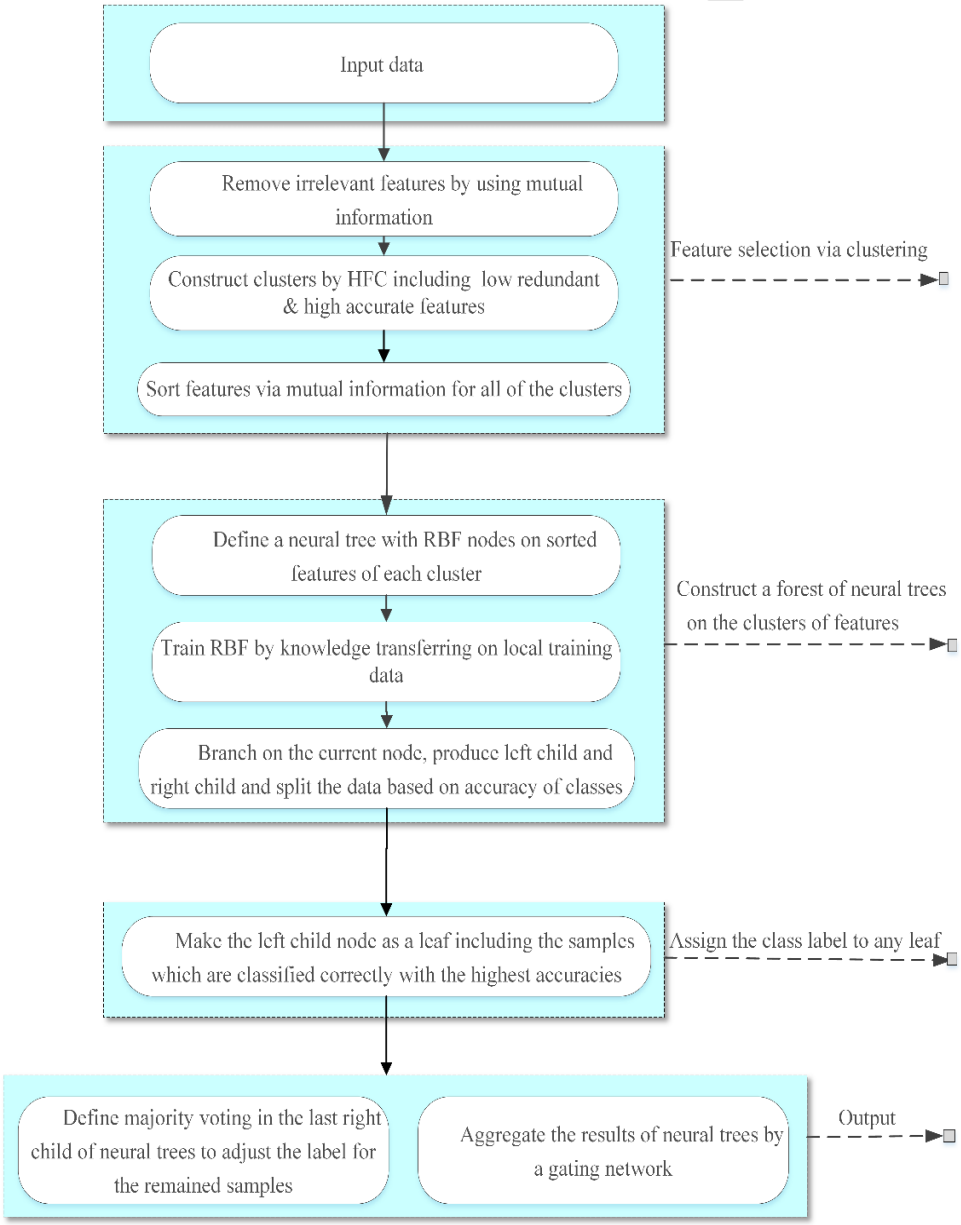


Figure 1 Main steps of the decision forest of RBF networks with HFC pre-processing

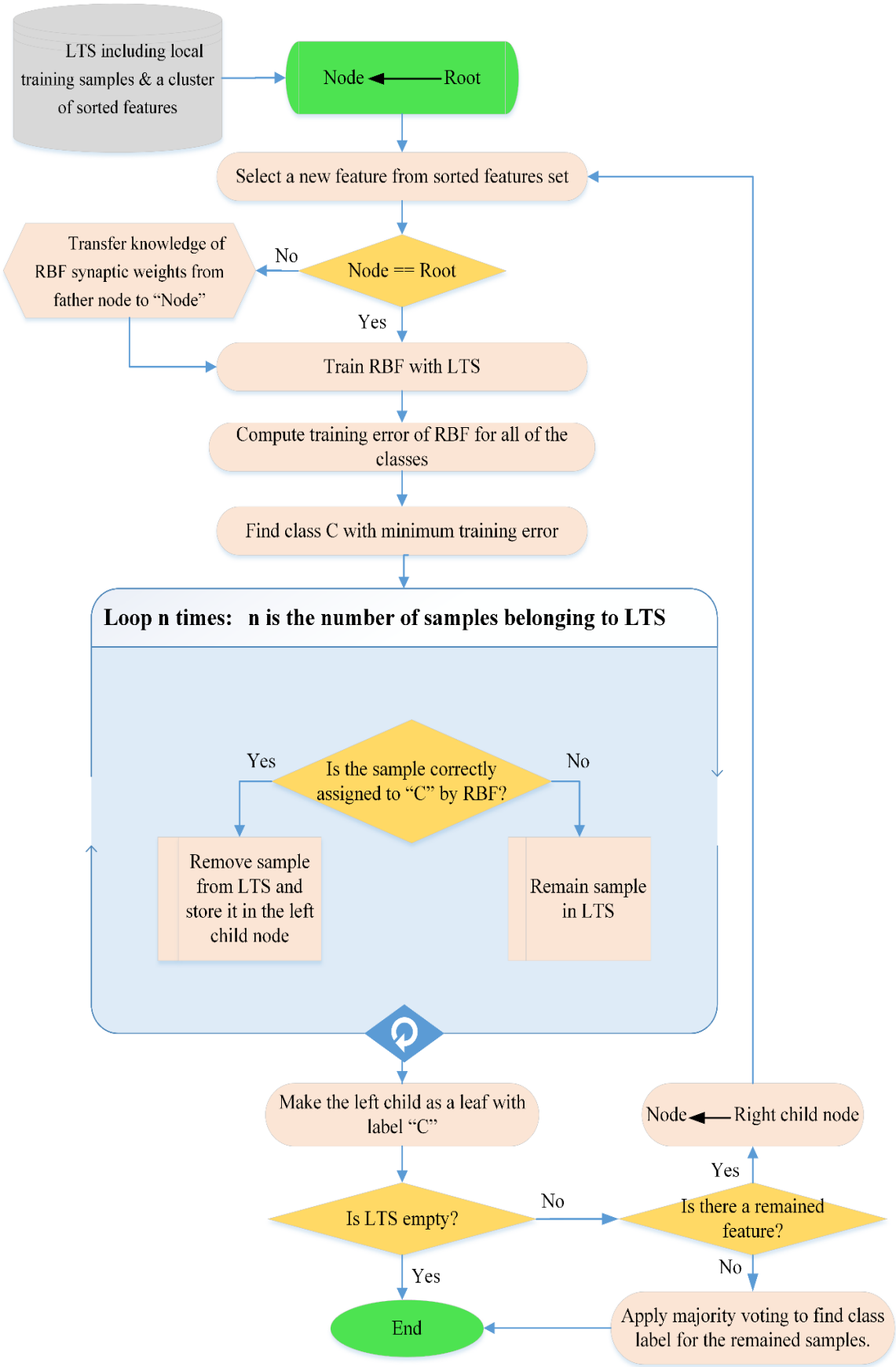


Figure 2 The process of a single neural tree with RBF networks in each cluster of FDRK

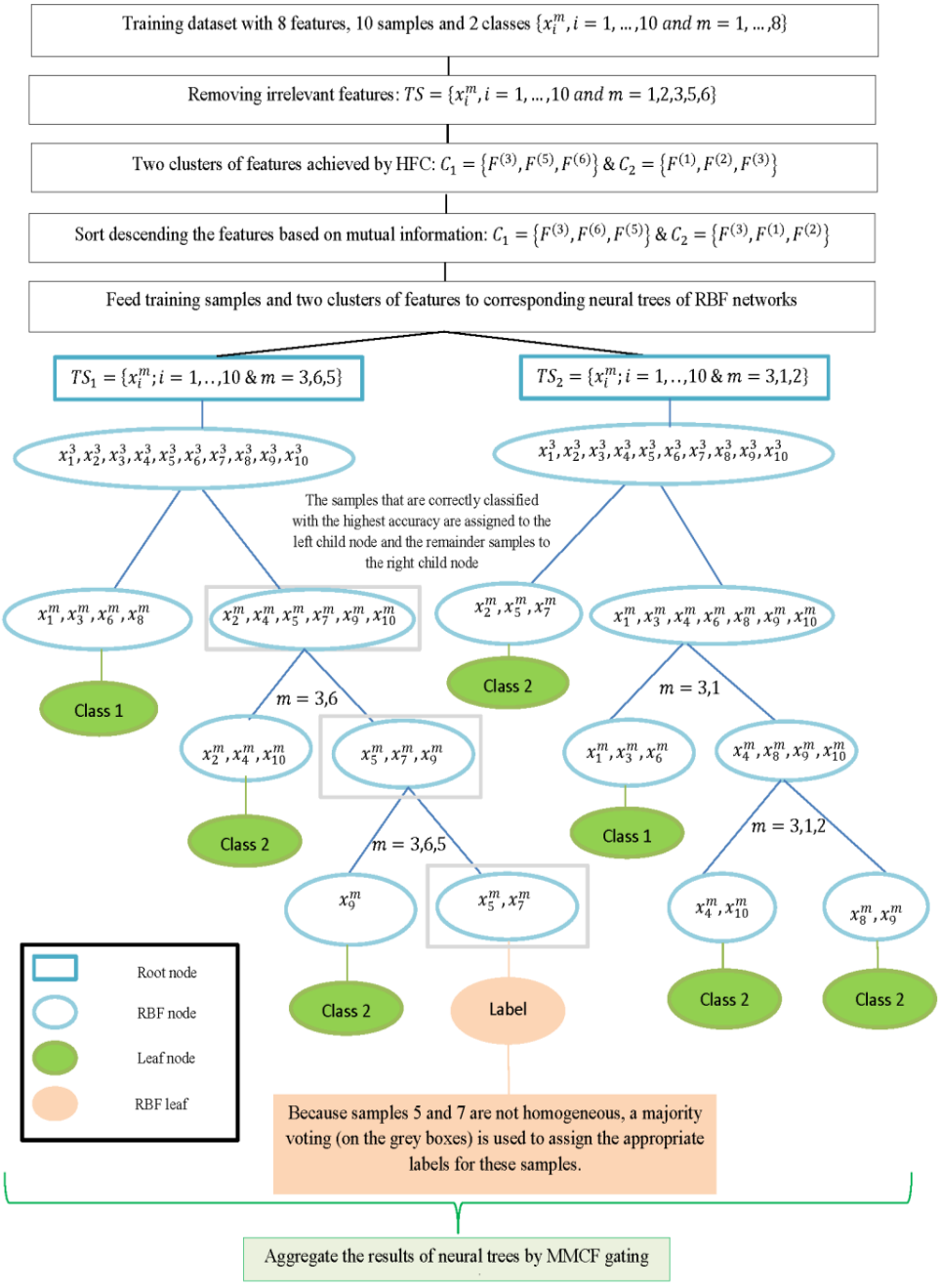


Figure 3 Example of the decision forest of RBF networks with HFC pre-processing

3 HFC steps

Both of the existing types of feature selection and the feature clustering methods usually eliminate some features that are dependent on target classes, because they have redundancies to others. This problem causes missing features. To overcome this drawback, HFC is proposed. In this method, we try to obtain some clusters of features such that the classification accuracy based on the

features of each cluster is high. In addition, to apply the most related features to target classes, we apply a new modified genetic algorithm (GA). HFC uses a filter approach to find the most related features to target classes. (Zare & Niazi, 2016) obtained that filter methods could work better on high-dimensional datasets and they are more rapid than the others. Therefore, a filter method with mutual information (MI) (Peng, et al., 2005) is considered. Really, MI is the amount of information that one variable could achieve in dealing with another variable. Two important applications of MI are finding irrelevant features (Zare & Niazi, 2016) and redundant features (Peng, et al., 2005). In former application, $\mathrm{MI}(\mathrm{x},\mathrm{y})$ compares the relevancy between each feature $\mathrm{x} \in \mathrm{X}$ in dealing with target value y. Therefore feature x, is irrelevant feature when $\mathrm{MI}(\mathrm{x},\mathrm{y})$ is low. In the latter application, $\mathrm{MI}(\mathrm{x}', \mathrm{x}^{"})$ could be used to find redundant features by computing MI across each pairs of features $\{\mathrm{x}', \mathrm{x}^{"}\} \subseteq \mathrm{X}$. For more information, one can refer to (Zare & Niazi, 2016), (Peng, et al., 2005) and (Vergara & Estévez, 2014). MI for discrete data can be computed as the following:

$$MI\ (x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} p\big(x(i),y(j)\big) \times log\left(\frac{p\big(x(i),y(j)\big)}{p\big(x(i)\big) \times p\big(y(j)\big)}\right). \quad (1)$$

In addition, for continuous data, MI can be computed based on (2), (Zare & Niazi, 2016), (Peng, et al., 2005).

$$MI\ (x,y) = \int_{x(i)}^{X}\int_{y(j)}^{Y} P\big(x(i),y(j)\big)\, log_2 \frac{P\big(x(i),y(j)\big)}{P\big(x(i)\big)P\big(y(j)\big)}\, dxdy, \quad (2)$$

where $\mathrm{P}\big(x(i)\big)$ is the occurrence probability of feature x and $\mathrm{P}\big(x(i),y(j)\big)$ is the joint probability when $\big(x(i),y(j)\big)$ occurs at the same time. Due to the computational complexity of the continuous version of MI and no closed form for the joint probability distribution, the discretization approach is recommended by (Peng, et al., 2005). This approach approximates the values of a function through a subinterval with a fixed value (Tóth, et al., 2008).

Based on the MI definition, we can filter the features by pursuing the following steps:

1- Calculate the MI for all of the features.
2- Compute the average and the standard deviation of MI values.
3- If MI of a feature is less than the average minus the standard deviation, remove this feature from feature space.

Based on the number of the remained features, we determine a threshold to limit the depth of the neural tree in each cluster. Note that, the time complexity of each neural tree is $O(hNM)$, where M is the number of features, N is the number of samples and h is the height of neural tree. To decrease time computation in each neural tree of FDRK, a predefined parameter ζ is defined and we try to meet $O(hNM) < \zeta$. With respect to this inequality, the number of features in each cluster can be defined. Then the clusters of the features are defined by applying GA. The goal of this GA is to maximize the accuracy of RBF networks and to minimize the redundancy, see Procedure 1. We consider the following basics to apply GA:

- The population of GA includes binary chromosomes. Any gen of a chromosome is 1 when the corresponding feature is selected in the chromosome. Each chromosome refers to a cluster of features. The number of gens with 1 is bounded by the maximum number of features satisfying $O(hNM) < \zeta$.

- For chromosome k, based on the selected features, the hybrid fitness function (3) is evaluated. By maximizing this fitness function in GA, we maximize RBF network accuracy as a wrapper feature selection method and minimize the inner redundancy as a filter one.

$$Fitness\ (k) = \frac{Accuracy\ (k)}{Redundancy\ (k)}. \tag{3}$$

- In (3), $Accuracy\ (k)$ is the training accuracy of RBF network in training set with respect to the corresponding features of cluster k and $Redundancy\ (k)$ is the normalized summation of MI values between all of the existing features in the cluster k, which can be computed by:

$$Redundancy\ (k) = \frac{1}{|cluster\ k|^2} \sum_{x',x^{"} \in\ cluster\ k} MI(x', x^{"}), \tag{4}$$

where $|cluster\ k|$ is the number of features in cluster k. Directly, $MI(x', x^{"})$ can be computed by substituting $(x', x^{"})$ instead of (x, y) in Equation (1). Note that the summation is gotten over all of the pairs of features of cluster k.

- For the crossover operator, as mentioned in Figure 4, two chromosomes with the least and the most fitness function values, are selected. Then, a gene location is selected randomly and these chromosomes are divided from this location and the binary values of these sub-chromosomes are changed. Then, if the number of selected features is greater than the maximum number of features, the algorithm refers to the mutation operator.
- For the mutation operator, if the number of selected features in a chromosome is less than the maximum number of features, some gens with zero value are selected randomly and they change to one. Also, if the number of selected features is greater than the maximum number of features, some gens with 1 value are selected randomly and they change to zero.
- For termination criterion, if the number of iterations is met, k chromosomes whose fitness values are greater than a predefined value are selected and the corresponding clusters are used to define FDRK.

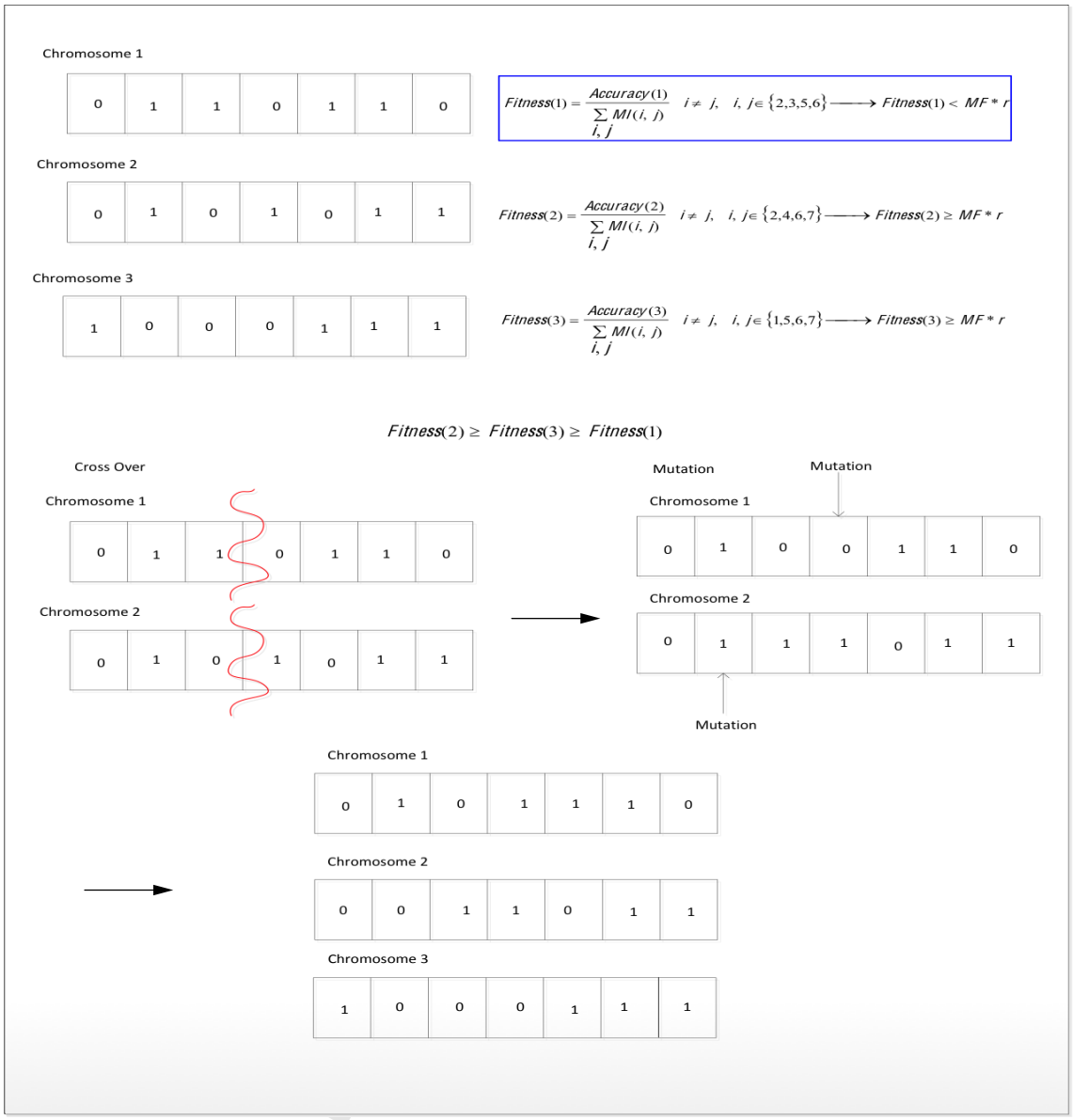


Figure 4 Feature Clustering by the modified GA (An example with 3 clusters with 4 selected features)

Procedure 1: Hybrid Feature Clustering (HFC)

Input

A problem with c classes and M features denoted with $FSET = \{F_m,\ m = 1, \dots, M\}$ where the learning sample denoted with $S = \{(x, y) | x = (x_1, x_2, \dots, x_M) \& \ y \in \{1, \dots, C\}\}$ including N samples.
$FNum$ is the upper bound on the number of features that a neural tree can process in a reasonable time.
GAI is the maximum iterations number of GA.
MF is a predefined threshold on the maximum value of the fitness function.

Output

Optimal clusters of features $FCluster = [\mathrm{F}^{(1)}, \dots, \mathrm{F}^{(K)}]$ where each $\mathrm{F}^{(i)}, i = 1, \dots, K$ consists of $FNum$ features.

Process

1. For $m = 1, \dots, \mathrm{M}$
   1.1 Compute MI $(\mathrm{x}_m, \mathrm{y})$ applying Eq. (1).
2. Compute the average ave and standard deviation θ of mutual information values MI $(\mathrm{x}_m, \mathrm{y}), \mathrm{m} = 1, \dots, \mathrm{M}$.
3. Set $FSUB = \{F_{m'},\ m' = 1, \dots, M'\}$, where MI $(\mathrm{x}_{m'}, \mathrm{y}) \geq (ave - \theta)$, MI $(\mathrm{x}_1, \mathrm{y}) \geq \cdots \geq$ MI $(\mathrm{x}_{m'}, \mathrm{y})$ and $M' \leq M$.
4. Set numbers of clusters as $= round\ (\frac{M'}{FNum})$.
5. Generate $CNum$ binary chromosomes, randomly, where each chromosome has $FNum$ gens with 1. Also, sort features of each cluster in descending order of MI values and set $r = 1$, $flag = 0$, $count = 0$ and $bestfit = 0$.
6. While $flag \neq 1$ do
   6.1 While $r < GAI$
      6.1.1 For $i = 1{:}\ CNum$
         6.1.1.1 Compute fitness function Fitness $_{(i)}$ for each cluster based on Eq. (3).
      6.1.2 Define $FitSort = \{Fitness_{(1)}, \dots, Fitness_{(CNum)}\}$, where $Fitness_{(1)} \geq \cdots \geq Fitness_{(CNum)}$.
      6.1.3 If $bestfit < Fitness_{(CNum)}$
         6.1.3.1 Set $bestfit = Fitness_{(CNum)}$ and $FCluster = [\mathrm{F}^{(1)}, \dots, \mathrm{F}^{(CNum)}]$
      6.1.4 Else go to 5.
      6.1.5 If $Fitness_{(1)} \geq (MF * r)$
         6.1.5.1 $count = count + 1$;
         6.1.5.2 For $k = 2{:}\ CNum$
            6.1.5.2.1 If $Fitness_{(k)} < (MF * r)$
               6.1.5.2.1.1 Do crossover on chromosomes 1 & k.
               6.1.5.2.1.2 For chromosomes 1 & k
                  If the number of gens with 1 value is greater than $FNum$, select some gens with 1 value randomly and change them to 0.
                  If the number of gens with 1 value is less than $FNum$, select some gens with 0 value randomly and change them to 1.
               6.1.5.2.1.3 Compute $Fitness_{(1)}$ and $Fitness_{(k)}$
            6.1.5.2.2 Else $count = count + 1$;
         6.1.5.3 If $count == CNum$
            Set $flag = 1$ & $FCluster = [\mathrm{F}^{(1)}, \dots, \mathrm{F}^{(CNum)}]$
         6.1.5.4 Else
            Set $r = r + 1$ and $count = 0$, Go to 6.1 .
7. Sort features in each cluster of $FCluster$ in descending order based on their mutual information.
8. Return $FCluster = [\mathrm{F}^{(1)}, \dots, \mathrm{F}^{(K)}]$, where $K = CNum$ and $\mathrm{F}^{(i)} = \{f_t, \dots, f_d\}$, where $\mathrm{F}^{(i)}$ has $FNum$ features($|\mathrm{F}^{(i)}| = FNum$), and MI $(f_t, \mathrm{y}) \geq \cdots \geq$ MI (f_d, y).

4 Components of FDRK

4.1 Decision forest of RBF networks

FDRK is a forest of neural trees whose nodes include RBF networks. Really, each of the neural trees classifies the samples with respect to the features covered by a cluster. Each neural tree does the following steps, which is more clarified in Procedure 2; see also Figure 2 for a graphical description about the process.

1. In the root node of each neural tree, the features of the corresponding cluster are sorted in descending order based on their MI values. In each level of a neural tree, the branching is done on a new feature with the highest MI in the sorted list.
2. Each node of neural tree consists of an RBF network to train samples with respect to the features that are selected till now. Each RBF network consists of a single layer and the numbers of nodes in the input and the hidden layers are equal to the features number. Centres of hidden nodes are selected once forever, based on k-mean clustering problem, which is solved in the root node of each neural tree. In addition, the number of nodes in the output layer is equal to the number of class labels.
3. For each RBF network in a father node namely n_l, the training error is evaluated for every class. Then a class with the least error is selected and the samples, which are assigned to this class correctly, are sent to the left child node and the other samples are sent to the right child node, as presented in Figure 2 and Figure 3. Since, the samples in the left child node are homogeneous, it becomes a leaf node and the branching continuous on the remainder samples covered by the right child node.
4. In the right child node, we use a strategy to transfer knowledge of father node (Abpeykar & Ghatee , 2018). This knowledge helps RBF network to train the samples in a reasonable time. For knowledge transferring, firstly, the features of the father node are transferred to the right child node, and then a new feature is added to them. The final weights of the RBF network of father node (denoted with $w(n_l)$) are transferred to the corresponding weights of the child node. For the weights corresponding to the new added feature, uniformly random numbers (denoted with $w(feature_{l+1})$) are defined. Now, for finding the optimal weights for RBF network of the child node, the vector $w(n_{l+1})$ consisting of vectors $w(n_l)$ and $w(feature_{l+1})$ is defined and then weight updating of sub-procedure 1 is used. For finding the optimal weights of any RBF network, it is possible to use pseudo-inverse technique (Braga, et al., 2002). As one can note that the complexity of determining pseudo-inverse for a matrix with M rows and N columns is $O(MN^3)$, see e.g., (Hwang & Bang, 1997). Thus, it is supposed to find the optimal weights of RBF by iterative methods in a less than a direct method such as pseudo-inverse. So, $Max_{Iteration}$ is considered as $Max_{Iteration} = MN^3$ in implementation of RBF training process presented in this subsection.

Sub-Procedure 1: Weight Updating

While $w(n_{l+1})$ doesn't converge and the iteration number is less than $Max_{Iteration}$)
{
$w(n_{l+1}) \leftarrow w(n_{l+1}) - \alpha \nabla E(n_{l+1})$;
Update the step size α and gradient value $\nabla E(n_{l+1})$;
}
If the loop terminates because of receiving to the number of $Max_{Iteration}$, similar to (Braga, et al., 2002), use pseudo-inverse method for finding the optimal $w(n_{l+1})$.

5. If all of the features of the cluster are used for classification, or the samples in right child node are homogeneous, go to the step 6, otherwise update the structure of neural tree, substitute $l \leftarrow l + 1$ and return to step 3.
6. In this step, a class label with the most majority should be assigned to all of the samples in the right child node and the algorithm terminates.

Procedure 2: Decision Forest of RBF Networks	
Input	A problem with c classes and M features denoted with $FSET = \{F_m,\ m = 1, \dots, M\}$ where the training sample denoted with $S = \{(x, y) \mid x = (x_1, x_2, \dots, x_M) \& \ y \in \{1, \dots, C\}\}$ including N samples. $FCluster = [\mathrm{F}^{(1)}, \dots, \mathrm{F}^{(K)}]$, where $K = CNum$ and $\mathrm{F}^{(i)} = \{f_t, \dots, f_d\}$, where $\mathrm{F}^{(i)}$ has $FNum$ features ($\left\lvert \mathrm{F}^{(i)} \right\rvert = FNum$), and MI $(f_t, \mathrm{y}) \geq \cdots \geq$ MI (f_d, y).
Output	Decision Forest of RBF Networks
Process	1. For $k = 1, \dots, \mathrm{K}$ 2. Assign $TS = \{(x, y) \mid x = (x_t, \dots, x_d) \& \ y \in \{1, \dots, c\}\}$ with features including $\mathrm{F}^{(k)} = \{f_t, \dots, f_d\}$ to the root node of k^{th} neural tree. 3. Set $i = 1$ & $Flag = 0$; 3.1. While $(Flag == 0)$ & *there is any unprocessed feature* of $\mathrm{F}^{(k)}$ 3.1.1. Select i^{th} feature of $\mathrm{F}^{(k)}$. 3.1.2. Assign $LTS = \{(x, y) \mid x = (x_I\ ; I = 1, \dots, i^{th} feature\ of\ \mathrm{F}^{(k)}) \ \& \ y \in \{1, \dots, C\}\}$ to the current node. 3.1.3. Train RBF network with samples belonging to LTS and update the synaptic weights by Sub-Procedure 1. 3.1.4. For each class $y \in \{1, \dots, C\}$ of LTS, compute the training error of the RBF network. 3.1.5. Find $c^* \in \{1, \dots, C\}$ as a class with the minimum training error. 3.1.6. For all samples of LTS do 3.1.6.1. If current sample correctly classified to class c^*, then assign current samples into the left child node. 3.1.6.2. Else, assign current sample into the right child node. 3.1.7. Label the left node with c^* as a leaf node. 3.1.8. If the right child node has not any sample, set $Flag = 1$; 3.1.9. Else 3.1.9.1. Set $i = i + 1$; 3.1.9.2. Substitute the right child node as the current node and assign samples of right node to LTS. 3.1.9.3. Transfer synaptic weights of father node to the same features of current node. 3.2. End of While. 4. If the last child node includes some samples without labels, a class label with the most majority should be assigned to all of the samples in the right child node. 5. Return Decision Forest of RBF Networks.

4.2 MMCF Gating Network

In the feature clustering phase, the clusters are not restricted to the disjoint sets. This means that they can cover some similar features. After this phase, for any new sample, its features is decomposed based on the features of each cluster and they are sent to the corresponding neural tree. The neural trees probably assign different label(s) to this sample. Therefore, it is necessary to use an appropriate gating network to aggregate the results. To develop a gating network, the following concepts are considered:

1). Aggregating the results of all of the neural trees to find an appropriate label for each given sample.
2). Using reliability-based aggregation techniques associate with majority voting when different labels exist for a given sample.

For this aim, MMCF is introduced that considers all of the leaves of FDRK to find the final class label. When all of the leaves assign a unique label for a sample, MMCF returns this class label and terminates. Otherwise, the majority voting is used. If this method does not assign a unique class label, the cheat method is used which refers to an RBF node with the best training performance. This gating mechanism also can be used in each neural tree of FDRK to decide regarding the appropriate label for all of the samples in the last right child node, when its samples are not homogenous. However, we have used just majority voting in the last right child node of all of the neural trees of FDRK.

5 Performance analysis of FDRK

FDRK performs well in both of computational time and accuracy. In what follows, the computational complexity of FDRK is presented using the similar ideas stated by (Savvas & Sofianidou, 2016). Then, some theoretical points are derived regarding the convergence of FDRK to a global optimum fitness value in each cluster.

5.1 Computational complexity analysis

The computational complexity of a neural tree depends on the number of nodes that contain RBF networks and the complexity of the RBF network. The RBF network tries to find optimal weight vector $\bar{W}$ and to assign the correct label for each input sample. Assume N is the number of samples, M is the number of features, n_{hidden} is the number of neurons of the hidden layer and C is the number of classes. Thus, the time complexity of assigning the input nodes to hidden nodes in a fully connected RBF network is $O(Mn_{hidden})$. In addition, the time complexity of assigning hidden nodes to output nodes is $O(n_{hidden}C)$. Therefore, the time complexity over all training samples is $O\big(N(Mn_{hidden}+n_{hidden}C)\big)$. In addition, in the last layer of the RBF network, the final weight vector should be found by an iterative method, which is presented in Sub-Procedure 1. But as it is presented in this subsection, its complexity is less than the complexity of a pseudo-inverse method which is equal to $O\big(Cn_{hidden}^3\big)$. Consequently, the time complexity of the RBF network can be given by $O\left(\big(Nn_{hidden}(M+C)\big)+\big(Cn_{hidden}^3\big)\right)$. For more information see e.g., (Hwang & Bang,

1997). By using the time complexity of RBF network and the number of nodes with RBF in each neural tree of FDRK, the time complexity of each of these trees can be stated as the following:

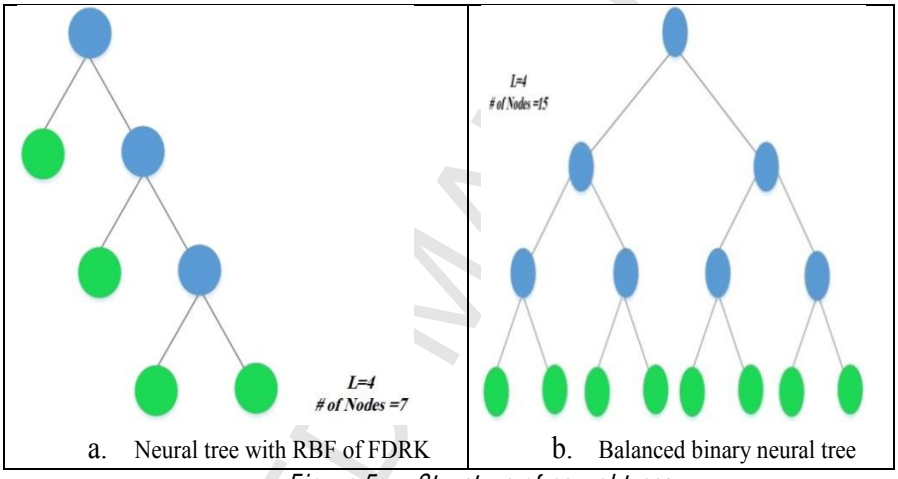
$$O\left(|RBF\ Nodes| \times \left(Nn_{hidden}(M+C)\right) + \left(Cn_{hidden}^{3}\right)\right), \tag{5}$$

where $|RBF\ Nodes|$ is the number of RBF nodes in the corresponding neural tree. To find $|RBF\ Nodes|$, as shown in Figure 5, for a constant height of the neural tree denoted by h, the number of nodes in a balanced binary neural tree (BBNT) is as follow:

$$|Node|_{BBNT} = 1 + 2 + 2^2 + .. + 2^{h-1} = 2^h - 1. \tag{6}$$

The number of nodes in each neural tree of FDRK is stated with (7), because in each level of neural tree, one node becomes a leaf node as shown in subfigure 'a' of Figure 5.

$$|Node|_{DRK} = 1 + 2 + 2 + .. + 2 = 2(h-1) + 1. \tag{7}$$



a. Neural tree with RBF of FDRK b. Balanced binary neural tree

Figure 5 Structure of neural trees

Therefore, the complexity of each neural tree in FDRK is better than the complexity of balanced binary neural tree, i.e., $O(h)$ which is the number of nodes in FDRK is better than $O(2^h)$ which is the number of nodes in the balanced binary neural trees. Since the latter is the most regular and efficient kind of neural trees in the literature (Micheloni, et al., 2012), FDRK is preferable on the other neural trees based on complexity viewpoint. Furthermore, subfigure 'a' of Figure 5 shows that the number of RBF nodes with blue colour is $|RBF\ Nodes| = h - 1$, which is equal to the height of the neural tree minus last node. In all of the nodes of neural trees, an RBF network with the above topology should be trained, therefore the terms $n_{hidden}C$ and Cn_{hidden}^{3} are constant in each neural tree of FDRK. Let $\lambda = Cn_{hidden}^{3}$. Since, λ is greater than n_{hidden} and Cn_{hidden}, the time complexity in each neural tree of FDRK can be simplified as the following:

$$O\big((h-1)(\lambda N(1+M)\big) = O(\lambda hNM). \tag{8}$$

Since λ is a fixed parameter, $O(\lambda hNM) = O(hNM)$. Besides, by using feature clustering, the height of each neural tree of FDRK denoted with h_k satisfies the following condition.

$$(C+1) \le h_k \le M, \tag{9}$$

where M is the total number of features of the training set and C is the number of classes. Note that the maximum number of features in a cluster is $M-1$. Thus, we can analyse the following cases:

(1) Minimum case: as the minimum height of a neural tree is $C+1$, therefore the minimum number of features which is equal to the number of nodes or with RBF network can be stated with c and the complexity of this neural tree is:

$$O(min-case) = O(C(C+1)N) = O(C^2N). \tag{10}$$

Therefore in the minimum case, if all of the neural trees have the minimum number of features the complexity of K neural tress that work on the clusters of features in FDRK is as the following:

$$O_{FDRK_{min}} = \sum_{k=1}^{K} O(C^2N) = K.O(C^2N) = O(KC^2N). \tag{11}$$

(2) Maximum case: the maximum number of features in each cluster is $M-1$. Therefore, the maximum height of neural tree is M and the time complexity is as follow:

$$O(max-case) = O\big(MN(M-1)\big) = O(M^2N). \tag{12}$$

When all of the neural trees have maximum height, the number of clusters is at most the combination of $(M-1)$ items from a collection with M items, i.e., $\binom{M}{M-1}$, which is equal to $\frac{M!}{(M-M+1)!(M-1)!} = M$ and the complexity of FDRK is as follow:

$$O_{FDRK_{max}} = \sum_{k=1}^{M} O(M^2N) = M.O(M^2N) = O(M^3N). \tag{13}$$

5.2 Convergence analysis

In this subsection, the convergence of FDRK is discussed. To study the error of FDRK as a classifier there exist different kinds of error functions including, L_1-norm (Least Absolute Error), L_2-norm (Mean Square Error) and the ratio of the number of misclassified samples over the total number of samples. The third one is used in experimental studies, since it is more common, while L_1-norm and L_2-norm are useful in theoretical contexts to analyse the error minimization problems (Hastie, et al., 2013). Table 1, compares these two norms in terms of optimization aspects (Hastie, et al., 2013).

Table 1. Differences between L_1-norm and L_2-norm

Beneficial aspects	L_1-norm	L_2-norm
Robust	*	
Stable solution		*
Unique solution		*
Built-in feature selection	*	
Sparsity	*	
Analytical solution		*

As this table shows, since $L_2 - norm$ leads to a unique and stable solution, it is used in the current study for convergence analysis. The total error is as the following:

$$Error = \sum_{c=1,\dots,C} Error_c, \tag{14}$$

where $Error_c$ is the error of class $c \in \{1, \dots, C\}$. Thus the error function can be stated as:

$$\begin{aligned} Error &= \frac{1}{N} \sum_{j=1,\dots,N} \left\| f(O(x_j)) - Y(x_j) \right\|^2 \\ &= \frac{1}{N} \sum_{j=1,\dots,N} \sum_{c=1,\dots,C} \left(f(O_c(x_j)) - Y_c(x_j) \right)^2, \end{aligned} \tag{15}$$

where N is the number of training samples, $Y(x_j) = [Y_1(x_j), \dots, Y_C(x_j)]$ and $O(x_j) = [O_1(x_j), \dots, O_C(x_j)]$ are the target vector and the output vector corresponding to x_j and f is a hard-limit activation function that returns 1 when its argument is greater than a threshold and returns zero, otherwise. Trivially x_j belongs to class c if and only if the c^{th} component of $Y(x_j)$ is one and the others are zero. Thus, Error counts the differences between two vectors $f(O(x_j))$ and $Y(x_j)$ which is an upper bound for misclassified samples. In addition, in each level l of FDRK, the output layer of RBF network minimizes the same error function to classify the samples:

$$Error_{RBF}(l) = \sum_{c=1,\dots,C} \frac{1}{N_{c,l}} \sum_{j=1,\dots,N_{c,l}} \left(f(O_c(x_j)) - Y_c(x_j) \right)^2, \tag{16}$$

where $N_{c,l}$ is the number of samples of class c in the node of level l and f is the activation function of the output layer which is also a hard-limit function. Also a linear activation function is considered in the input layer and a radial bases function R is used in the hidden neurons. The synaptic weights between the input layer and the hidden layer are considered equal to 1 and the synaptic weight between the hidden neuron i and the output neuron c is denoted with $w_{i,c}$. Then for each $c \in \{1, \dots, C\}$, we can write:

$$O_c(x_j) = \sum_{i \in HiddenNeurons} w_{i,c} O_i^H(x_j), \tag{17}$$

where $O_i^H(x_j)$ is the output of hidden neuron i and can be stated as the following:

$$O_i^H(x_j) = R\left(\sum_{f \in Features} x_{j,f} \right), \tag{18}$$

in which $x_{j,f}$ is the f^{th} component (feature) of sample x_j and $R()$ is a radial bases function which is used in hidden neurons. Therefore, the total error of each neural tree on cluster k of FDRK, denoted with $Error_{Cluster}(k)$, is defined as the following summation of error values over all levels:

$$\begin{aligned} Error_{Cluster}(k) &= \sum_l Error_{RBF}(l) \\ &= \sum_l \sum_{c=1,\dots,C} \frac{1}{N_{c,l}} \sum_{j=1,\dots,N_{c,l}} \left(f(O_c(x_j)) - Y_c(x_j) \right)^2. \end{aligned} \tag{19}$$

By getting the deviation from this error function with respect to $w_{i,c}$, the following is concluded:

$$\frac{\partial Error_{Cluster}(k)}{\partial w_{i,c}} = \sum_{l} \sum_{c=1,\dots,C} \frac{1}{N_{c,l}} \sum_{j=1,\dots,N_{c,l}} \frac{\partial \left(f(O_c(x_j)) - Y_c(x_j)\right)^2}{\partial w_{i,c}} \tag{20}$$

$$= \sum_{l} \sum_{c=1,\dots,C} \frac{1}{N_{c,l}} \sum_{j=1,\dots,N_{c,l}} \frac{\partial \left(f(O_c(x_j)) - Y_c(x_j)\right)^2}{\partial O_c(x_j)} . \frac{\partial O_c(x_j)}{\partial w_{i,c}}$$

$$= \sum_{l} \sum_{c=1,\dots,C} \frac{1}{N_{c,l}} \sum_{j=1,\dots,N_{c,l}} 2\left(f(O_c(x_j)) - Y_c(x_j)\right) . \frac{\partial f(O_c(x_j))}{\partial O_c(x_j)} O_i^H(x_j).$$

Since $f(O_c(x_j))$ is not differentiable, it is approximate with a linear function and so $\frac{\partial f(O_c(x_j))}{\partial O_c(x_j)} = constant$. Without loss of generality it is assumed as 1. Thus, following equation can be reached:

$$\frac{\partial Error_{Cluster}(k)}{\partial w_{i,c}} \tag{21}$$

$$\cong \sum_{l} \sum_{c=1,\dots,C} \frac{1}{N_{c,l}} \sum_{j=1,\dots,N_{c,l}} 2\left(f(O_c(x_j)) - Y_c(x_j)\right) . O_i^H(x_j)$$

$$= \sum_{l} \sum_{c=1,\dots,C} \frac{1}{N_{c,l}} \sum_{j=1,\dots,N_{c,l}} 2\left(f(O_c(x_j)) - Y_c(x_j)\right) . R\Big(\sum_{f \in Features} x_{j,f} \Big).$$

Thus, to improve the synaptic weights, by choosing positive step-size α, the following is resulted:

$$w_{i,c}^{new} = w_{i,c}^{old} - \alpha \frac{\partial Error_{Cluster}(k)}{\partial w_{i,c}} \tag{22}$$

$$= w_{i,c}^{old} - \alpha \sum_{l} 2\left(f(O_c(x_j)) - Y_c(x_j)\right) . R\left(\sum_{f \in Features} x_{j,f} \right).$$

Since any radial function R produces positive value for any input, the above derivative direction depends just on the sign of $f(O_c(x_j)) - Y_c(x_j)$. This means that if the output of neuron c with respect to sample x_j is different from observed value $Y_c(x_j)$, the synaptic weight is adjusted by this term. Thus, the convergence in cluster happens when for all levels of neural tree in this cluster we can converge $f(O_c(x_j)) - Y_c(x_j)$ to zero. Since in each level of FDRK the samples of a single class are recognized and are assigned to a leaf node, $f(O_c(x_j)) - Y_c(x_j)$ for these samples converge to zero and the other samples are sent to the next levels for error minimization. Thus, the convergence over all $f(O_c(x_j)) - Y_c(x_j)$ is guaranteed and so the iterative formula in (22) leads to convergence to the optimal weights in the successive levels of the neural tree in the corresponding cluster. On the other hand, based on Sub-Procedure 1, if the computation steps of Equation (22) increases, the method transforms to pseudo-inverse calculation to converge $f(O_c(x_j)) - Y_c(x_j)$ to zero. Thus the complexity of this part is less than or equal to the complexity of pseudo-inverse

computation which is polynomial, i.e., $O(n^{\omega})$ where ω is a constant dependent on the matrix multiplication, see e.g., (Gao & Antsaklis, 1991).

Since the knowledge transferring is just used for initialization of the synaptic weights, $w_{i,c}$ in the RBF of each node is not dependent on the other RBF networks in the different levels of neural tree. This means that the optimization over any RBF does not destroy the optimality of the others and thus by Equation (22), all of the weights are optimized separately. Also, the error function is quadratic convex function and thus a local optimal in each RBF node is also a global optimal. This completes the proof of the following theorem.

Theorem 1. Applying FDRK, the cluster error $Error_{Cluster}(k)$ converges to a global solution in a polynomial complexity time.

Assume that $fitness\left(x_{k'}^{(r)}(i)\right)$ denotes the value of fitness function of GA in the chromosome (or cluster) k' and state i at generation r. The following lemma is met.

Lemma 1. Let $Z_r = max\{fitness(x_{k'}^{(r)}(i))|k' = 1,\cdots,K'\}$ *be a sequence of the best solutions of GA for generation* r. *GA converges to the global optimum "*$fitness^*$*" if and only if* $lim_{r\to\infty} P\{Z_r = fitness^*\} = 1$, *(see e.g.* (Ming, et al., 2006)*).*

Theorem 2. Let both of the cluster size $|\text{cluster}\, k'|$; $k' \in \{1,\cdots,K'\}$ and generation number r converge to infinity. Then, GA receives to a cluster k such that its fitness function ($fitness\left(x_k^{(r)}(i)\right)$) converges to the global optimum of error in the last level of FDRK.

Proof:

By Theorem 1, the final level of FDRK converges to a global optimum weight of RBF networks and the error function is minimized. On the other hand, for high-dimensional datasets, the size of each cluster is very large. Since $Error_{Cluster}(k')$, $k' = 1,\cdots,K'$ is the global minimum of the neural tree k', and the $Error_{Cluster}(k')$ is the summation of error of all RBF networks in the corresponding neural tree, the training accuracy of RBF networks in each cluster k' are maximum.

In addition, when $|\text{cluster}\, k'|$ converges to infinity, the redundancy of cluster k' converges to minimum.

$$Redundancy\,(k') = \lim_{|\text{cluster}\, k'|\to\infty} \frac{1}{|\text{cluster}\, k'|^2} \sum_{x',x'' \in \text{cluster}\, k'} \text{MI}(x', x'') \to 0. \tag{23}$$

Therefore, as Accuracy(k') converges to the maximum and Redundancy (k') converges to the minimum, the fitness function $fitness\left(x_{k'}^{(r)}(i)\right)$ converges to maximum. In addition, in the GA procedure, is it tried to achieve the $fitness\left(x_{k'}^{(r)}(i)\right) \geq (\text{MF} * \text{r})$ for each cluster $k' = 1,\cdots,K'$. When the iterations on GA increases ($\text{r} \to \infty$) and by searching between all maximum finesses values, it could be expected that there is a $\text{k} \in \{1,\cdots,K'\}$ such that:

$$\lim_{\text{r}\to\infty} \text{P}\{\text{Z}_\text{r} = fitness^*\} = 1. \tag{24}$$

This means that in r^{th} generation, we receive $fitness\left(x_k^{(r)}(i)\right) = fitness^*$. Based on Lemma 1, there is a cluster $\text{k} \in \{1,\cdots,K'\}$ that converges to the global maximum.■

6 Experimental Study

Ensemble methods are powerful classifiers in dealing with high dimensional datasets. Data partitioning, using different classifiers in each partition and making decision based on aggregating the results of different classifiers are the reasons of their good performances. Among these methods, FDRK firstly uses a genetic algorithm to find some appropriate clusters of features and in each level recognizes a single class with great certainty and sends its samples to the left child node. However, for the classes with less certainty, the node does not judge and sends them to the next level for more investigation. Therefore, the reasons of achieving good classification performance by FDRK can be summarized as the following:

- Before last level, each RBF network involves to recognize a single class similar to one-vs-rest classifiers.
- In the last level, a great opportunity exists for the remainder samples to recognize by an accumulate knowledge achieved by multiple RBF networks which are trained in the previous levels. This opportunity corrects some false responses, which may be happened in the previous networks.
- Considering multiple neural trees in FDRK provides another opportunity for all of the samples to recognize by different experts and the final decision for each sample is extracted from an accumulated knowledge collected by multiple neural trees as the experts.

To explain the first opportunity, note that for non-classified samples in the final node, each neural tree of FDRK aggregates the results of all of the RBF nodes by using majority voting between RBF networks. In the second opportunity, MMCF gating network is used to assign a label to a given sample. In this scheme, firstly a majority voting method is used on neural trees and if it doesn't determine a unique class label, a cheat method is used. This cheat method considers the decision made by the most accurate neural tree. Thus, by these two successive modifications, FDRK improves the results of classification. In this section, the good classification performance of FDRK will be studied across different experiments.

Since, the proposed ensemble method in this paper, consists of HFC, FDRK and MMCF, in the first experiment, three different gating networks are compared with MMCF. Since in this experiment MMCF shows the best results, it is used in the next experiments of this section. In addition, the results of HFC are compared with the other feature selection methods to prove the benefit of keeping some features in HFC compared with the methods which remove these features. Then, we show that FDRK classifies the samples better than the previous neural trees. Finally, the good results of the proposed classification system on high-dimensional data will be reported together with the analysis on noisy situations. The details of the experimental benchmarks are presented in Appendix A. In addition, in Table 3-Table 5, the results of the other methods are completely copied and for fair analysis the same experimental setup is used for FDRK.

6.1 Comparison with different gating networks

To determine the effectiveness of MMCF for FDRK classifier, a comparison is done with three popular gating networks, including cheat, vote and average method (Rani, et al., 2015). Note that:

- In cheat method, the RBF network with the highest accuracy in FDRK, classifies the new incoming samples.

- In vote method, a majority voting is done on the results of RBF networks in FDRK.
- In average method, the class label of a new incoming sample is the average of RBF networks in FDRK.

Table 2, shows the results across different datasets with 3-fold and 10-fold cross validation. As this table shows, the average results of MMCF in most cases are better than the other gating networks. In addition, it obtains 100% accuracy in most of the cases, because it uses the most accurate RBF for classification in its final level. MMCF results are also similar to the cheat method for some datasets; however it provides better results in other datasets. Thus, MMCF is more powerful and it will be used in the next experiments.

Table 2. Different gating networks compared with MMCF with 10-fold cross validation

Data	Other Gating Networks			MMCF	
	Cheat	Vote	Average	3-fold CV	10-fold CV
IRIS	100.00	100.00	100.00	100.00	100.00
Ecoli	100.00	100.00	100.00	100.00	100.00
Glass	100.00	100.00	100.00	100.00	100.00
Breast	92.85	90.95	88.56	81.42	92.85
Wine	100.00	99.21	95.28	100.00	97.23
Heart	100.00	94.36	90.25	100.00	87.50
Letter	99.36	96.78	99.64	93.50	100.00
Vehicle	98.42	81.49	88.57	100.00	100.00
Hepatitis	90.30	87.23	85.54	98.20	96.84
Segment	98.00	95.00	95.83	98.25	100.00
WDBC	94.73	93.56	93.85	100.00	100.00
Ionosphere	95.23	94.28	92.53	96.28	96.28
Dermatology	99.07	88.88	90.58	98.25	99.07
Satellite	100.00	100.00	100.00	100.00	100.00
Wave	95.39	94.21	94.89	96.54	95.33
Spmabase	99.57	96.26	95.28	98.70	100.00
Sonar	93.54	85.43	84.61	91.66	91.66
DNA	95.92	90.17	91.21	94.36	96.45
Numerals	100.00	100.00	100.00	100.00	100.00
Hill	100.00	95.28	93.87	96.39	100.00
Arrhythmia	92.26	90.79	92.41	92.37	94.21
Lung	90.21	90.16	90.22	91.74	90.26
Madelon	91.81	90.84	91.25	92.88	92.88
Colon	100.00	98.26	97.31	100.00	100.00
Lym	93.29	92.11	92.67	92.29	94.32
Gisette	100.00	100.00	98.57	100.00	99.09
NCI	93.22	93.18	92.99	94.39	92.58
Arcene	100.00	100.00	100.00	100.00	100.00
Cll	93.75	93.26	91.11	94.26	92.21
Dexter	95.27	93.29	91.77	96.57	98.56
Dorothea	98.69	98.22	96.13	97.23	99.28
Gas1	88.28	84.42	81.66	88.35	85.31
Pems	88.27	86.97	83.25	85.38	89.76
Gas 2	82.97	82.14	80.20	82.29	84.37
Twin gas	84.95	83.50	82.88	85.57	83.26
URL	79.27	77.39	76.49	80.96	79.52

6.2 Comparison with different classifiers

In this subsection FDRK is compared with other methods on some famous datasets, which have been considered in the related literature. Since there is not a unique reference to present the best results of different classifiers on different datasets and to achieve more consistency in the context, the accuracy of FDRK with 3-fold and 10-fold cross validation on different datasets are

presented in Figure 6. Then, in each of the experiments, the results on some of datasets, which have been presented in the references are emphasized.

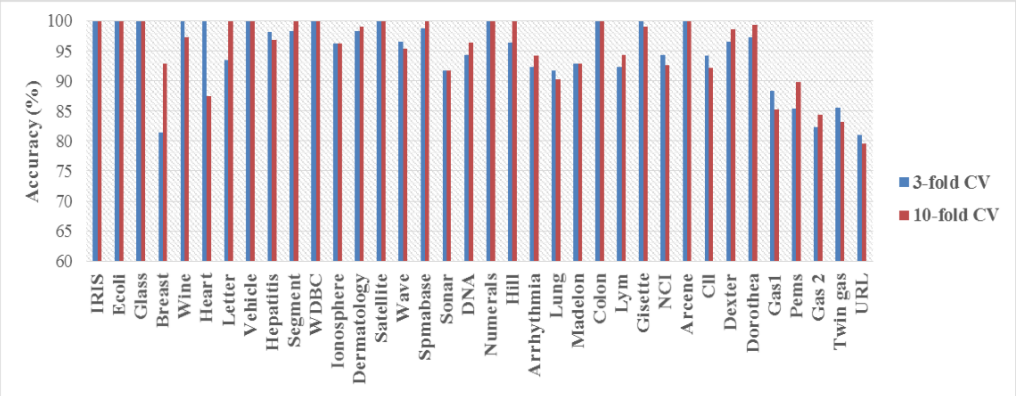


Figure 6 Accuracy of FDRK in 3-fold and 10-fold cross validation

- HFC versus other feature selection methods

In this subsection, the classification performance of the GA-based HFC method is compared with some existing filter and wrapper methods. Some of the previous works based on the other heuristic methods, such as ant colony optimization and practical swarm optimization are also considered in this experiment. Note that some of the feature selection methods remove some important features and just a unique subset of the features or some effective features remain for the classification purpose. However, by the proposed HFC, some of these eliminated features are kept in the clusters based on their high dependencies on class labels and minimum redundancies with other existing features in that cluster. Therefore, by comparing the results of HFC and the other feature selection methods, we can show that keeping these features in the non-redundant clusters improves the performance of FDRK.

Table 3 shows the comparison results of seven filter feature selection methods in WEKA on different datasets applying different classifiers including decision tree, random forest, Naïve Bayes (NB), SVM, and kNN. These classifiers are used for comparison because many references used them to examine the performance of the classifiers, see e.g., (Moradi & Rostami, 2015). The results on GCACO (Moradi & Rostami, 2015), L-Score (He, et al., 2005), F-Score (Gu, et al., 2012), RRFS (Ferreira & Figueiredo, 2012), mRMR (Peng, et al., 2005), ReliefF (Robnik-Sikonja & Kononenko, 2003) and UFSACO (Eroglu & Kilic, 2017), have been presented by (Moradi & Rostami, 2015) and their maximum results are copied in the first column of all classifiers in Table 3. The second column for each classifier show the result found by HFC. This table shows that the results of HFC are better than other classifiers except of "Hepatitis" and "Ionosphere" with decision tree and, "Sonar" with SVM. However, HFC with FDRK overcomes all of the classifiers in these datasets, as mentioned in the last column of Table 3. The weak performance of HFC in "Hepatitis" happens because there is a single cluster in this dataset and HFC has not any effect. In addition, the results of HFC in high-dimensional datasets including "Madelon", "Colon" and "Arcene" are better than the other feature selection methods and its accuracy is greater than 85%, which is an important result on the high-dimensional datasets. Also, note that FDRK classifier achieves leaves in all of levels of neural trees; this made the results very interesting. Finally, the results show that keeping the highly related features in the non-redundant clusters increases the efficiency of the classification process.

In addition, HFC is compared with four wrapper feature selection methods including, GCACO (Moradi & Rostami, 2015), PSOFS (Xue, et al., 2013), ACOFS (Kabir, et al., 2012) and, HGAFS (Kabir, et al., 2011) by using SVM and NB classifiers in WEKA. The maximum accuracies of these algorithms are retrieved from (Moradi & Rostami, 2015) and have been presented in Table 4. HFC works better than four wrapper methods except of NB for "WDBC" dataset and SVM for "Sonar" dataset. In addition, HFC with FDRK gets the best results in all experiments. The reduction in the accuracy of HFC with NB and SVM in these two datasets may be happened because of the structure of the datasets, which may fit with NB and SVM but not with RBF.

Table 3. Maximum classification accuracy of GCACO (Moradi & Rostami, 2015), L-Score (He, et al., 2005), FScore (Gu, et al., 2012), RRFS (Ferreira & Figueiredo, 2012), mRMR (Peng, et al., 2005), ReliefF (Robnik-Sikonja & Kononenko, 2003), UFSACO (Eroglu & Kilic, 2017) as filter feature selection methods and the average classification accuracy of HFC based on DT, RF SVM, kNN and NB algorithms

Classifier	Decision Tree			Random Forest			NB			kNN			SVM			HFC+FDRK		
Dataset	Best Reported Features Selector	Max. accuracy of previous methods	Average accuracy of HFC+DT	Best Reported Features Selector	Max. accuracy of previous methods	Average accuracy of HFC+RF	Best Reported Features Selector	Max. accuracy of previous methods	Average accuracy of HFC+NB	Best Reported Features Selector	Max. accuracy of previous methods	Average accuracy of HFC+kNN	Best Reported Features Selector	Max. accuracy of previous methods	Average accuracy of HFC+SVM	3-fold CV	10-fold CV	Number of clusters
Wine	GCACO	93.76	95.82	GCACO	96.19	96.72	GCACO	95.73	95.80	RRFS	94.42	96.66	RRFS	96.55	96.65	100.00	97.23	2
Hepatitis	GCACO	84.33	77.75	mRMR	83.81	77.35	GCACO	84.71	86.79	GCACO	85.84	86.25	GCACO	84.52	86.31	98.20	96.84	1
WDBC	mRMR	93.98	94.01	GCACO	95.36	96.11	mRMR	93.67	94.33	mRMR	92.58	95.03	mRMR	94.24	95.27	100.00	100.00	2
Ionosphere	GCACO	89.74	88.23	GCACO	90.76	91.89	GCACO	90.24	90.71	GCACO	89.40	91.22	GCACO	90.41	91.76	96.28	96.28	2
Spambase	GCACO	89.21	90.15	GCACO	89.19	92.26	GCACO	88.22	88.99	GCACO	88.94	90.21	GCACO	88.38	89.56	98.70	100.00	2
Sonar	GCACO	79.57	89.60	GCACO	81.43	87.40	GCACO	77.60	88.80	GCACO	80.41	81.43	GCACO	82.38	81.89	91.66	91.66	4
Arrhythmia	GCACO	60.38	63.63	GCACO	61.58	66.23	GCACO	60.38	67.50	GCACO	60.25	62.26	GCACO	60.51	63.13	92.37	94.21	11
Madelon	GCACO	82.73	88.01	F-Score	77.93	89.21	GCACO	79.32	87.98	GCACO	74.82	88.45	GCACO	78.42	89.22	91.74	90.26	34
Colon	GCACO	80.00	87.03	GCACO	82.28	86.33	GCACO	79.04	90.02	GCACO	80.47	88.90	GCACO	81.42	89.23	92.88	92.88	125
Arcene	GCACO	67.24	91.00	GCACO	71.45	92.02	GCACO	68.94	91.04	UFSACO	66.78	91.03	GCACO	68.38	91.28	94.39	92.58	650

Table 4. Maximum of average of classification accuracy of GCACO (Moradi & Rostami, 2015), PSOFS (Xue, et al., 2013), ACOFS (Kabir, et al., 2012) and HGAFS (Kabir, et al., 2011) wrapper feature selection methods and the average classification accuracy of HFC using SVM and NB algorithms

Classifier	NB			SVM			HFC+FDRK		
Datasets	Best Reported Features Selector	Max. accuracy of previous methods	Average accuracy of HFC+NB	Best Reported Features Selector	Max. accuracy of previous methods	Average accuracy of HFC+SVM	3-fold CV	10-fold CV	Number of Clusters
Wine	GCACO	95.73	95.80	PSOFS	94.42	96.65	100.00	97.23	2
Hepatitis	GCACO	84.71	86.79	GCACO	84.52	86.31	98.20	96.84	1
WDBC	PSOFS	94.69	94.33	PSOFS	94.81	95.27	100.00	100.00	2
Ionosphere	GCACO	90.24	90.71	PSOFS	90.75	91.76	96.28	96.28	2
Spambase	GCACO	88.22	88.99	ACOFS	87.30	89.56	98.70	100.00	2
Sonar	HGAFS	80.70	88.80	GCACO	82.38	81.89	91.66	91.66	4
Arrhythmia	ACOFS	63.11	67.50	PSOFS	62.52	63.13	92.37	94.21	11
Madelon	GCACO	79.32	90.00	PSOFS	80.06	89.22	91.74	90.26	34
Colon	PSOFS	83.33	90.02	PSOFS	85.23	89.23	92.88	92.88	125
Arcene	PSOFS	69.78	91.04	PSOFS	71.42	91.28	94.39	92.58	650

Furthermore, Figure 7 shows the percentage of perfectness of the feature clustering applying ACO, PSO, the standard GA and the modified GA as the heuristic methods across 10 datasets. This figure reveals that the results of our modified GA almost in half part of experiments provide better results compared with the other heuristic methods.

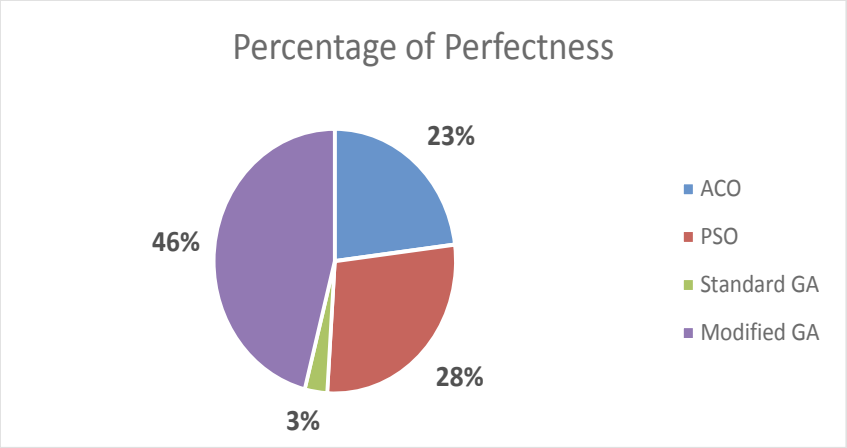


Figure 7 Percentage of perfectness of ACO, PSO, standard GA and modified GA feature clustering methods combining with SVM and NB classifiers

- FDRK versus other neural trees

In this subsection, FDRK is compared with six other neural trees across 14 datasets as mentioned in Table 5. The maximum results of other neural trees have been presented in the first columns of this table where some of the results are based on 3-fold cross validation and others on 10-fold cross validation. Besides, the average results of FDRK in 3-fold and 10-fold cross validation are illustrated in the last two columns. FDRK gets better accuracy than other neural trees, except in "Breast" dataset in which the performance of COF-NT is greater than FDRK. As it is shown in the final column of Table 5, the number of feature clusters in the "Breast" dataset is one. Therefore, the forest consists of just a single neural tree and GA could not perform well to find the best clusters.

Table 5. Comparison between the accuracy of other neural trees and FDRK with MMCF gating network

Datasets	COF-NT (Rani, et al., 2015)	BNT (Micheloni, et al., 2012)	NT-SLP (Foresti & Pieroni, 1998)	NT-MLP (Maji, 2008)	SLNN (Fontenla-Romero, et al., 2010)	Random Forest (Hall, et al., 2009)	Decision Tree	FDRK+MMCF	
								3-fold CV	10-fold CV
IRIS	98.09	96.07	96.05	97.33	96.12	95.33	94.1	100.00	100.00
Ecoli	82.90	82.42	82.33	81.71	80.41	82.44	80.36	100.00	100.00
Glass	72.23	70.23	69.23	73.10	69.29	66.60	-	100.00	100.00
Breast	96.56	94.86	94.23	93.27	95.63	96.13	-	81.42	92.85
Heart	80.15	78.12	78.81	79.10	72.23	78.14	-	100.00	87.50
Letter	87.50	83.99	84.82	82.90	77.33	93.29	86.6	93.50	100.00
Vehicle	82.67	78.98	79.88	81.10	76.89	77.04	-	100.00	100.00
Segment	93.76	91.47	90.51	95.06	85.21	96.39	-	98.25	100.00
Ionosphere	94.76	94.16	94.04	91.16	86.60	92.80	91.1	96.28	96.28
Dermatology	95.14	94.02	89.94	94.72	91.06	94.05	-	98.25	99.07
Satellite	86.90	84.01	82.60	86.10	79.58	86.01	85.2	100.00	100.00
Wave	81.19	79.11	81.14	80.56	80.11	81.11	-	96.54	95.33
DNA	95.11	93.16	90.52	94.2	92.29	89.96	93.3	94.36	96.45
Hill	60.03	58.91	58.03	57.24	57.13	60.56	-	96.39	100.00

6.3 Comparison with other ensembles

In this subsection, the ensemble of HFC+FDRK+MMCF is compared with the previous ensemble methods including AdaBoostM1 (Eibl & Pfeiffer, 2002), Baggign (Ting & Witten, 1997), Dagging (Ting & Witten, 1997), LogitBoost (Frank, et al., 2002), MODLEM (Stefanowski, 1998), Decorate (Melville & Mooney, 2003), Grading (Seewald & Fürnkranz, 2001), MultiBoostAB (Webb, 2000) and StackingC (Seewald, 2002). The results of these ensembles are provided by WEKA and are presented in Table 6. This table shows that the accuracy of other ensembles are close to the accuracy of our proposed method in datasets with small numbers of features. However, the results of these methods on datasets with high-dimensional features are worse than the results of the proposed ensemble method. Note that, the other ensembles get results in high-dimensional datasets in more than 30 minutes, and in some cases, they take three hours to get the results. Thus to present a fair comparison, after 30 minutes, if the ensemble could not classify the samples, we terminated it and present a dash in Table 6. Nevertheless, the maximum process time of our proposed ensemble method is about three minutes. The only case that the proposed ensemble method gets worse results than the previous ensembles is in the "Breast" dataset, where the proposed ensemble method achieves just one cluster. In addition, other ensembles are not reliable in all datasets and they get high accurate result in some cases and low accurate results in the others. However, the results of FDRK ensemble method in different datasets even with large number of features are completely acceptable. This happens, because each level of the neural trees of FDRK contains a leaf node and therefore a single class can be recognized in every level with great confidence.

Table 6. Comparison between HFC+FDRK+MMCF and other ensemble methods

Datasets	AdaBoostM1	Bagging	Dagging	LogitBoost	MODLEM	Decorate	Grading	MultiBoostAB	StackingC	HFC+FDRK+MMCF	
										3-fold CV	10-fold CV
IRIS	95.33	94.00	86.00	94.00	96.00	94.66	33.33	94.00	33.33	100.00	100.00
Ecoli	66.66	85.62	66.66	87.76	65.13	--	43.73	66.66	43.73	100.00	100.00
Glass	44.85	72.42	56.42	71.49	--	91.96	35.51	44.85	35.51	100.00	100.00
Breast	94.84	96.42	94.99	95.70	94.84	96.42	65.52	94.99	65.52	81.42	92.85
Wine	91.57	94.94	85.95	96.31	--	96.62	39.88	85.95	39.88	100.00	97.23
Heart	83.33	81.88	78.88	83.70	81.11	--	55.55	82.96	55.55	100.00	87.50
Letter	7.07	81.26	7.67	64.88	84.12	--	4.06	7.07	4.06	93.50	100.00
Vehicle	70.02	91.66	70.02	86.68	96.12	93.40	70.02	70.02	70.02	100.00	100.00
Hepatitis	82.58	81.29	83.22	81.93	80.64	83.22	79.35	82.58	79.35	98.20	96.84
Segment	30.40	95.86	59.80	95.86	--	97.40	15.73	30.40	15.73	98.25	100.00
WDBC	94.55	95.43	92.97	95.95	--	96.48	62.74	93.67	62.74	100.00	100.00
Ionosphere	90.88	91.16	86.03	91.16	91.73	92.02	64.10	84.04	64.10	96.28	96.28
Dermatology	50.27	93.16	54.64	96.17	95.35	97.54	30.60	50.27	30.60	98.25	99.07
Satellite	38.95	86.40	57.90	84.75	84.95	88.50	23.50	38.95	23.50	100.00	100.00
Wave	50.74	80.00	50.86	82.62	81.82	--	33.84	50.74	33.84	96.54	95.33
Spmabase	90.06	94.00	82.93	92.00	94.11	94.52	60.59	84.43	60.59	98.70	100.00
Sonar	71.63	76.92	69.71	79.32	70.67	84.13	53.36	74.51	53.36	91.66	91.66
DNA	86.73	52.53	72.00	94.95	34.73	93.98	51.88	75.51	51.88	94.36	96.45
Numerals	19.20	73.50	38.65	78.45	76.90	78.20	10.00	19.20	10.00	100.00	100.00
Hill	50.49	50.24	50.49	50.49	--	--	50.49	50.49	50.49	96.39	100.00
Arrhythmia	55.53	72.12	55.30	74.55	--	66.81	54.20	55.75	54.20	92.37	94.21
Lung	78.12	81.25	75.00	81.25	62.50	78.12	71.87	71.87	71.85	91.74	90.26
Madelon	63.40	75.05	57.25	63.02	52.52	73.35	50.10	61.75	50.10	92.88	92.88
Colon	74.19	79.03	58.06	77.41	62.90	82.25	64.51	87.09	64.51	100.00	100.00
Lym	74.32	76.35	75.67	83.10	76.35	81.75	54.72	73.64	54.79	92.29	94.32
Gisette	88.95	75.08	82.23	89.43	--	82.21	48.13	82.77	48.13	100.00	99.09
NCI	--	--	--	--	--	--	--	--	--	94.39	92.58
Arcene	79.50	82.50	74.50	85.50	86.00	--	56.00	80.00	56.00	100.00	100.00
Cll	--	--	--	--	--	--	--	--	--	94.26	92.21
Dexter	78.12	81.25	75.00	81.25	62.50	78.12	71.87	71.87	71.87	96.57	98.56
Dorothea	73.43	72.33	65.23	73.06	68.06	78.23	38.23	77.48	35.67	97.23	99.28
Gas1	--	--	--	--	--	--	--	--	--	88.35	85.31
Pems	--	--	--	--	--	--	--	--	--	85.38	89.76
Gas 2	--	--	--	--	--	--	--	--	--	82.29	84.37
Twin gas	--	--	--	--	--	--	--	--	--	85.57	83.26
URL	--	--	--	--	--	--	--	--	--	80.96	79.52

6.4 Comparison on artificially generated data

In this subsection, a controlled simulation study is made on artificially generated datasets. To this aim, some samples are generated from a multivariate normal distribution with different means $\boldsymbol{\mu_i}, i = 1,2,$ for two classes with assumed equal identity covariance matrices. As a well-known result in statistical decision theory, the Bayes classifier attains the minimum classification error among the others by assuming known features distribution and the class prior distributions (Hastie, et al., 2013). The efficiency of FDRK along with the Bayes classifier based on Gaussian distributions as well as other benchmark methods namely RBF network, decision tree, support vector machine (SVM) and ensemble methods Bagging, LogitBoost and Decorate are examined across the following three scenarios:

- Scenario 1: Benchmarks with 1000 samples and 100 features.
- Scenario 2: Benchmarks with 1000 samples and 500 features.
- Scenario 3: Benchmarks with 1000 samples and 1000 features.

In each scenario, the mean vectors $\boldsymbol{\mu}_i, i = 1, 2$, are designed to yield different overlapping between classes in a range of low, median and high overlaps based on (Ho & Basu, 2002). More overlapping between classes results in more complexity in classification problem and lower accuracy. Table 7 presents the details of the results along different scenarios. The FDRK performs better than the other benchmark methods on these different experimental scenarios except of Bayes classifier. In addition, as expected the Bayes classifier attains the best results among the others according to the ideal simulation settings. However, the FDRK provide close results to Bayes classifier even in dataset with high overlap between classes, which verifies the strength of the proposed approach. Moreover, the simulation results show that the superiority of the FDRK than the other ensemble based techniques.

Table 7. Comparison on artificially generated datasets(Bold values show the best classifier and FDRK results)

Artificially generated datasets	Comparison Measure	Classifier	Low overlap	Median overlap	High overlap
1000 samples with 100 features	Train Accuracy	RBF network	78.52	48.22	27.93
		Decision Tree	77.25	44.38	28.59
		Bayes Classifier	**92.76**	**61.21**	**52.86**
		SVM	59.45	40.23	24.39
		Bagging	66.32	42.26	40.90
		LogitBoost	55.97	40.72	28.39
		Decorate	56.56	46.75	27.93
		FDRK	**90.02**	**61.10**	**51.53**
	Test Accuracy	RBF network	75.21	44.68	25.00
		Decision Tree	74.63	42.10	26.39
		Bayes Classifier	**92.76**	**61.21**	**52.86**
		SVM	56.11	32.16	20.06
		Bagging	62.85	40.29	38.65
		LogitBoost	52.11	38.92	28.00
		Decorate	52.39	44.56	25.15
		FDRK	**89.66**	**60.33**	**51.00**
1000 samples with 500 features	Train Accuracy	RBF network	84.32	51.92	30.62
		Decision Tree	80.00	48.24	32.22
		Bayes Classifier	**99.98**	**74.81**	**56.78**
		SVM	64.54	46.28	28.00
		Bagging	69.92	51.76	47.11
		LogitBoost	59.74	44.45	32.56
		Decorate	58.92	50.01	33.27
		FDRK	**99.33**	**55.65**	**52.98**
	Test Accuracy	RBF network	81.25	50.01	28.63
		Decision Tree	78.65	46.44	30.25
		Bayes Classifier	**99.98**	**74.81**	**56.78**
		SVM	62.25	44.20	28.00
		Bagging	69.54	50.28	45.85
		LogitBoost	58.00	43.52	30.01
		Decorate	57.46	49.21	30.00
		FDRK	**98.65**	**53.00**	**51.33**
1000 samples with 1000 features	Train Accuracy	RBF network	86.56	52.91	33.50
		Decision Tree	85.11	50.23	36.85
		Bayes Classifier	**100.00**	**94.25**	**84.80**
		SVM	67.56	44.23	28.50
		Bagging	73.21	53.65	48.12
		LogitBoost	64.78	50.22	38.54
		Decorate	64.90	52.21	36.70
		FDRK	**99.97**	**70.66**	**63.66**
	Test Accuracy	RBF network	85.50	51.99	33.22
		Decision Tree	85.01	50.11	36.56
		Bayes Classifier	**100.00**	**94.25**	**84.80**
		SVM	66.52	42.21	28.50
		Bagging	70.00	52.96	47.00
		LogitBoost	62.21	48.55	35.96
		Decorate	60.11	52.00	35.44
		FDRK	**99.95**	**84.21**	**63.42**

6.5 Comparison on high-dimensional datasets

In this subsection, some datasets with large number of features are examined by FDRK. In Table 8, the benchmarks with greater than 10000 features are mentioned. Based on this table, the average of accuracies of FDRK is 90.16%. Especially, the average of accuracies for datasets with 10000 features to 100000 features is 97.26% and the average of accuracies for datasets from 100000 features to 500000 features is 85.53%. The results for datasets with 2 classes are very high; however, for other datasets with more classes the lowest accuracy is still 82.29%, which is noticeable.

As it is shown in this table, this method could efficiently deal with datasets when the number of features is great compared with the number of samples. This occurs because each cluster contains some non-redundant features and the samples are repeated in all of the neural trees of the forest. Therefore, a small set of samples could be trained by concerning different clusters of features in these neural trees and this approach improves the classification accuracy by using multiple perspectives dedicated by multiple neural trees. To validate this evidence, the "URL Reputation" dataset can be considered, which is a very big dataset with more than 2 million samples and more than 3 million features. Using FDRK on this dataset, 79.52% accuracy is attained, which is very promising but not as a perfect result compared with the others. Because the rate of features number to samples number is close to one and FDRK cannot help us by classification regarding to multiple perspectives of different neural trees. However, the accuracy of FDRK in this dataset is still acceptable; because traditional decision tree and RBF network cannot classify this dataset even in 1 hour computation time and a single neural tree with RBF nodes classify this dataset with 54.89% accuracy.

Table 8. Classification accuracy of datasets with more than 10000 features

Datasets	# of features	# of samples	# of classes	(#features)/ (#samples)	# of clusters	Accuracy of single neural tree with RBF Nodes	Accuracy of FDRK	
							3-fold CV	10-fold CV
Arcene	10000	900	2	11	650	78.31	100.00	100.00
Cll	11340	111	3	102	720	68.72	94.26	92.21
Dexter	20000	2600	2	8	1200	84.44	96.57	98.56
Dorothea	100000	1950	2	51	4670	76.44	97.23	99.28
Gas 1	120432	58	12	2076	5321	68.31	88.35	85.31
Pems	138673	440	7	315	5689	64.23	85.38	89.76
Gas 2	150000	180	10	833	5949	68.4	82.29	84.37
Twin gas	480000	640	10	750	6231	48.31	85.57	83.26
URL	3231961	2396130	2	1	89321	54.89	80.96	79.52

In addition, as it was mentioned before, there are two opportunities which could improve the classification performance of FDRK by using majority voting and MMCF gating network. To validate these two claims, a new experiment is done on high-dimensional benchmarks. This experiment tries to evaluate the effects of the presented opportunities on the performance of FDRK. Therefore, the following two variations are considered:

1. FDRK without any gating network (In this case instead of multiple neural trees with RBF networks, a single neural tree without any gating network in its last level is applied).

2. FDRK with both of the majority voting in the last level of each neural tree and gating network to aggregate the results of neural trees of FDRK.

Figure 8 presents the results of these two scenarios considering different benchmarks. As one can see, for all of the benchmarks, the results of the second variation using gating network are surprising compared with FDRK without gating network. In addition, it could be seen that by increasing the number of features the accuracy of the FDRK in the second variation improves noticeably.

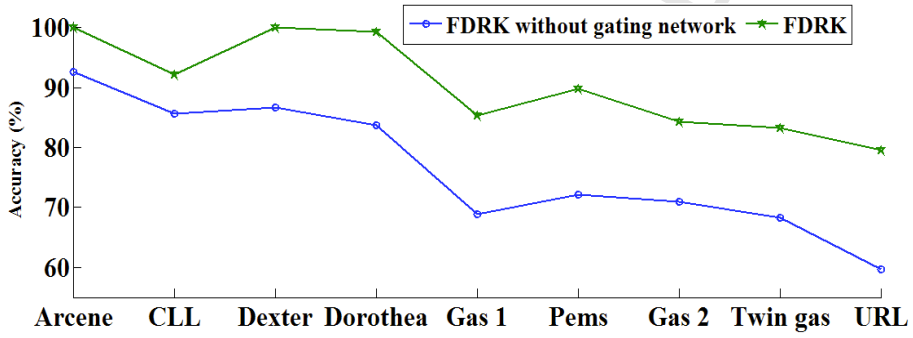


Figure 8 Analyzing the effect of MMCF to improve the performance of FDRK

6.6 Comparison under noisy conditions

In this subsection, the effect of noise on FDRK method is studied. In each of the datasets, 5%, 15% and 25% of Gaussian noise are influenced. The sensitivity analysis of noise effect on results along with the noiseless ones is performed. In all of the experiments, FDRK runs 100 times. Figure 9 represents the average of results in these 100 runs. As one can see when the rate of noise increases, the performance of FDRK decreases significantly. However, this figure illustrates that the proposed forest even under the noisy conditions, provides reasonable results and the accuracies in the datasets with high-dimensional features are still greater than 70%. Therefore, FDRK is almost robust in noisy environment even in high dimensional datasets.

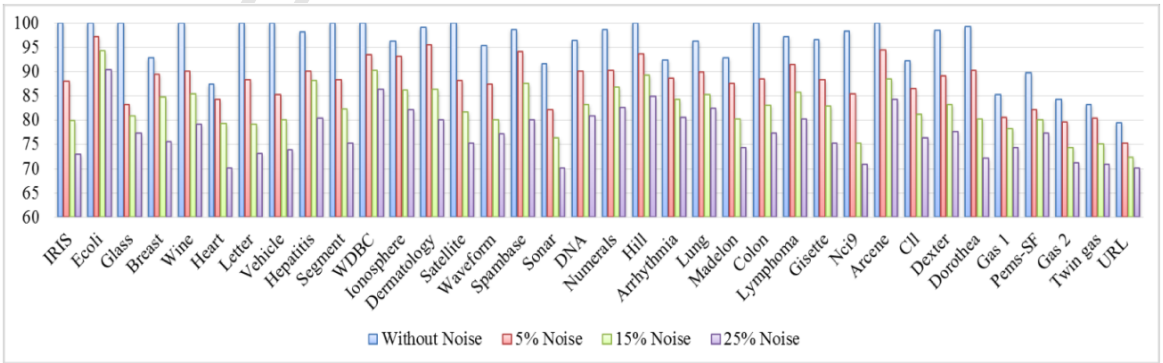


Figure 9 Comparison between accuracy of FDRK under different levels of Gaussian noise

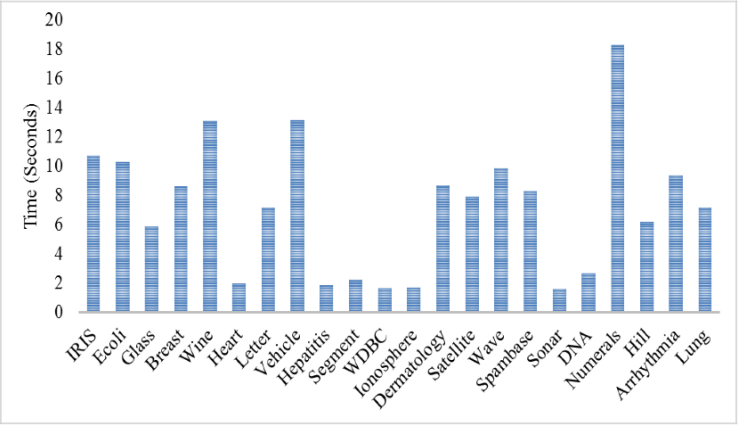
6.7 Comparison on processing time of GA

A modified GA is defined as a preprocessing phase before FDRK training process. This phase implements once on each benchmark. Therefore, its computational cost cannot effect on the training process of FDRK. However, three issues are interesting to note and to investigate.

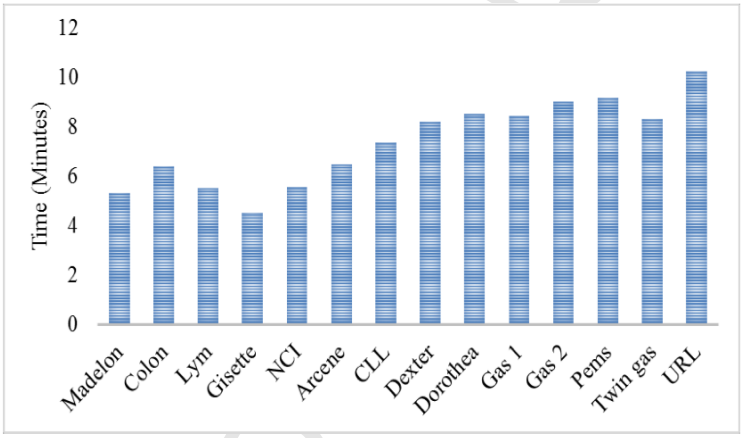
1. The process time of GA for all benchmarks in our study is presented in Figure 10. As one can see, in subfigure 'a' the process time of GA is small because the number of features is less than 500 but for datasets with more features, subfigure 'b' shows the computational time is between 4 and 13 minutes.

2. There are many references regarding the metaheuristic algorithms for feature selection, feature extraction and feature clustering, see e.g., (Xue, et al., 2016) and (Iglesia, 2013). They are out of the scope of this paper, because this paper tries to develop a new architecture for ensemble learning. However, it is a direct work to pursue such powerful metaheuristic methods as a preprocessing phase to cluster the features and then the proposed architecture in this paper can be used on these clusters to classify high-dimensional datasets.

3. It is an interesting problem to cluster the features by a hybrid of descent gradient method and some probabilistic methods for redundancy analysis (Ghalwash, et al., 2016). Such methods can be considered in the next researches to cluster the features with more accuracy and when the pre-processing phase of FDRK gets better clusters, we hope to improve the performance of FDRK for some benchmarks that mentioned in the current paper and we cannot classify them perfectly.



a) Processing time (in second) for GA to define feature clustering on datasets with low-dimensional features



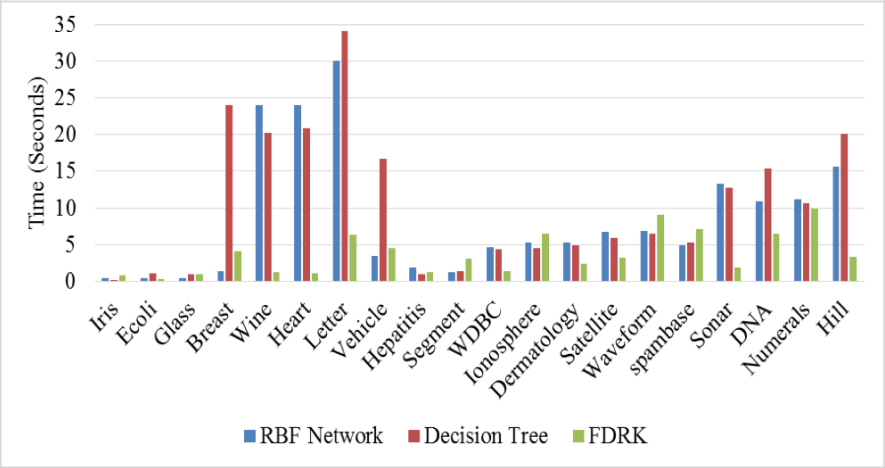
b) Processing time (in minute) for GA to define feature clustering on datasets with high-dimensional features

Figure 10 Processing time of GA for feature clustering

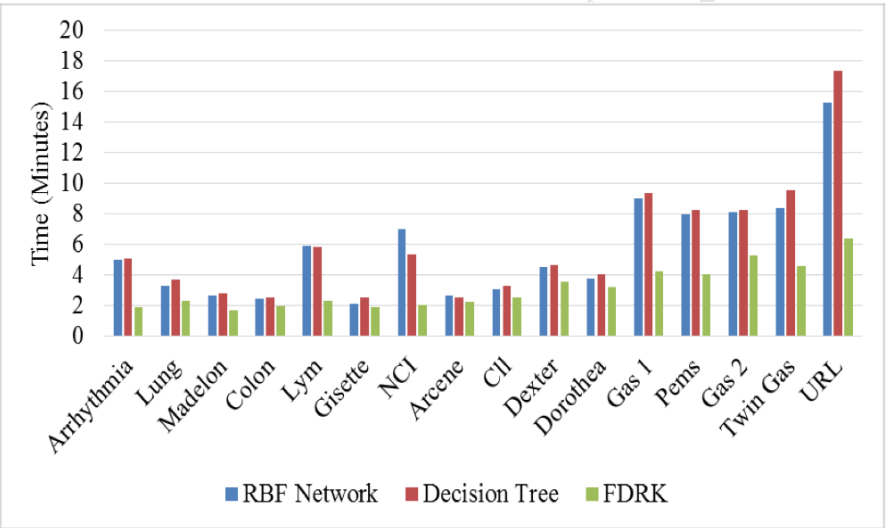
6.8 Comparison on processing time of FDRK

In this subsection, we examine the processing time of FDRK. Figure 11 shows that the highest processing time of FDRK on datasets with low dimensional and high-dimensional features are 9.06 seconds and 6.34 minutes, respectively. However, the computational times of the previous works have not been presented. Therefore, this parameter can't be compared with them, but it seems that these computational times are acceptable in real situations. Furthermore, the process times of the simple RBF network and the decision tree are good in some cases, but their accuracies are less than FDRK as it is discussed before.

In addition, by putting together these results with results of Subsection 6.7, one can say that the processing time of FDRK is small and HFC algorithm consumes the greatest part of processing time in our ensemble method. But HFC is a pre-processing phase and it could not make great effect on the perceived time of our proposed ensemble method, which consists of HFC, FDRK and MMCF. This means that, the processing time of our ensemble method is still acceptable.



a. Processing time of FDRK on dataset with less than 100 features



b. Processing time of FDRK on dataset with more than 100 features

Figure 11 Comparison between the computational time of FDRK, Decision Tree and RBF

7 Conclusion

This paper introduces an ensemble method to classify the data with high-dimensional features. First, a Hybrid Feature Clustering method called HFC is proposed to define a good clustering of features. For this aim, a modified version of Genetic Algorithm (GA) with a hybrid fitness function is developed. Each of these feature clusters contains the minimum inner redundancy and the maximum training accuracy. Then a forest of several neural trees namely FDRK is developed where neural trees classify the samples by considering the features covered by clusters of features. In the nodes of neural trees, RBF networks are used as classifier. Finally, a gating network aggregates all of the results of neural trees of the forest. In experiments, it is shown that a Mix gating network based on Majority vote and Cheat method (MMCF) provides the best results for FDRK classifier.

The feature clustering phase was examined with seven filters and four wrapper feature selection methods, based on 10 different datasets, and in almost all of the experiments, HFC, got

the best accuracy. In addition, feature clustering based on the modified GA gains better results, compared with the feature clustering methods based on other heuristics, including ACO, PSO and standard GA. To examine the performance of FDRK, this method was compared with previous neural trees on 14 datasets. In 13 out of 14 datasets, the proposed method gets better results.

To compare the efficiency of the proposed method on high-dimensional datasets, FDRK is applied on some datasets with 100 to more than 3,000,000 features and in all of them, the performance of the ensemble method of HFC, FDRK and MMCF was better than the other methods. In addition, FDRK is compared with the other ensembles, and it is shown that the proposed method outperforms the state-of-the-art methods even in high-dimensional features. Besides, the complexity of FDRK is analyzed and its convergence is proved theoretically. In addition, the noise analysis shows that the noise could not disturb the performance of FDRK, and so this classifier is almost robust in noisy environment.

References

Abpeykar S, Ghatee M., 2018. An ensemble of RBF neural networks in decision tree structure with knowledge transferring to accelerate multi-classification. Neural Computing and Applications.:1-21.

Anuradha, J., 2015. A brief introduction on Big Data 5Vs characteristics and Hadoop technology. *Procedia computer science*, *48*, pp.319-324.

Bak, B.A. and Jensen, J.L., 2016. High dimensional classifiers in the imbalanced case. *Computational Statistics & Data Analysis*, *98*, pp.46-59.

Braga, A., Carvalho, A. C., Ludermir, T., de Almeida, M., & Lacerda, E., 2002. Radial Basis Functions Networks. In *Modelling and Forecasting Financial Data* (pp. 159-178). Springer, Boston, MA.

Chen, X.W. and Lin, X., 2014. Big data deep learning: challenges and perspectives. *IEEE access*, *2*, pp.514-525.

De La Iglesia, B., 2013. Evolutionary computation for feature selection in classification problems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *3*(6), pp.381-407.

Dernoncourt, D., Hanczar, B. and Zucker, J.D., 2014. Analysis of feature selection stability on high dimension and small sample data. *Computational statistics & data analysis*, *71*, pp.681-693.

Eibl, G. and Pfeiffer, K.P., 2002, August. How to make AdaBoost. M1 work for weak base classifiers by changing only one line of the code. In *European Conference on Machine Learning* (pp. 72-83). Springer, Berlin, Heidelberg.

Eroglu, D.Y. and Kilic, K., 2017. A novel Hybrid Genetic Local Search Algorithm for feature selection and weighting with an application in strategic decision making in innovation management. Information Sciences, 405, pp.18-32.

Ferreira, A.J. and Figueiredo, M.A., 2012. An unsupervised approach to feature discretization and selection. *Pattern Recognition*, *45*(9), pp.3048-3060.

Fontenla-Romero, O., Guijarro-Berdiñas, B., Pérez-Sánchez, B. and Alonso-Betanzos, A., 2010. A new convex objective function for the supervised learning of single-layer neural networks. *Pattern Recognition*, *43*(5), pp.1984-1992.

Foresti, G. L., &Pieroni, G., 1998. Exploiting neural trees in range image understanding. Pattern Recognition Letters, 19(9), pp.869-878.

Frank, E., Holmes, G., Kirkby, R. and Hall, M., 2002, November. Racing committees for large datasets. In *International Conference on Discovery Science* (pp. 153-164). Springer, Berlin, Heidelberg.

Frénay, B., Doquire, G. and Verleysen, M., 2014. Estimating mutual information for feature selection in the presence of label noise. Computational Statistics & Data Analysis, 71, pp.832-848.

Gao, Z. and Antsaklis, P.J., 1991. Stability of the pseudo-inverse method for reconfigurable control systems. *International Journal of Control*, *53*(3), pp.717-729.

García-Torres, M., Gómez-Vela, F., Melián-Batista, B. and Moreno-Vega, J.M., 2016. High-dimensional feature selection via feature grouping: A Variable Neighborhood Search approach. *Information Sciences*, *326*, pp.102-118.

Gayathri, C., 2014. Feature subset selection using filtering with Mutual information and Maximal information coefficient. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(1), pp.1350-1354.

Ghalwash, M.F., Cao, X.H., Stojkovic, I. and Obradovic, Z., 2016. Structured feature selection using coordinate descent optimization. *BMC bioinformatics*, *17*(1), p.158.

Goswami, S., Das, A.K., Chakrabarti, A. and Chakraborty, B., 2017. A feature cluster taxonomy based feature selection technique. *Expert Systems with Applications*, *79*, pp.76-89.

Gu, Q., Li, Z. and Han, J., 2012. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*.

Gupta, P., Jain, S. and Jain, A., 2014. A review of fast clustering-based feature subset selection algorithm. *International Journal of Scientific & Technology Research*, *3*(11), pp.86-91.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H., 2009. The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), pp.10-18.

Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A. and Khan, S.U., 2015. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, *47*, pp.98-115.

Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A. and Khan, S.U., 2015. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, *47*, pp.98-115.

Hastie, T., Tibshirani, R. and Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Corr. 7th printing 2013 edition. New York, NY: Springer.

He, X., Cai, D., &Niyogi, P. (2005, December). Laplacian score for feature selection. In NIPS (Vol. 186, p. 189).

Ho, T.K. and Basu, M., 2002. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (3), pp.289-300.

Hwang, Y.S. and Bang, S.Y., 1997. An efficient method to construct a radial basis function neural network classifier. Neural networks, 10(8), pp.1495-1503.

Kabir, M.M., Shahjahan, M. and Murase, K., 2011. A new local search based hybrid genetic algorithm for feature selection. *Neurocomputing*, *74*(17), pp.2914-2928.

Kabir, M.M., Shahjahan, M. and Murase, K., 2012. A new hybrid ant colony optimization algorithm for feature selection. *Expert Systems with Applications*, *39*(3), pp.3747-3763.

Krier, C., François, D., Rossi, F. and Verleysen, M., 2007, April. Feature clustering and mutual information for the selection of variables in spectral data. In ESANN (pp. 157-162).

Laney, D., 2001. 3D data management: Controlling data volume, velocity and variety. META group research note, 6(70), p.1.

Li, J. and Liu, H., 2017. Challenges of feature selection for big data analytics. IEEE Intelligent Systems, 32(2), pp.9-15.

Liu, H. and Yu, L., 2005. Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on knowledge and data engineering, 17(4), pp.491-502.

Mahani, A.S. and Sharabiani, M.T., 2015. SIMD parallel MCMC sampling with applications for big-data Bayesian analytics. Computational Statistics & Data Analysis, 88, pp.75-99.

Maji, P., 2008. Efficient design of neural network tree using a new splitting criterion. Neurocomputing, 71(4-6), pp.787-800.

Melville, P. and Mooney, R.J., 2003, August. Constructing diverse classifier ensembles using artificial training examples. In *IJCAI* (Vol. 3, pp. 505-510).

Micheloni, C., Rani, A., Kumar, S., &Foresti, G. L., 2012. A balanced neural tree for pattern classification. Neural Networks, 27, pp.81-90.

Ming, L., Wang, Y. and Cheung, Y.M., 2006, July. On convergence rate of a class of genetic algorithms. In *Automation Congress, 2006. WAC'06. World* (pp. 1-6). IEEE.

Moradi, P. and Rostami, M., 2015. Integration of graph clustering with ant colony optimization for feature selection. *Knowledge-Based Systems*, *84*, pp.144-161.

Napoli, C., Pappalardo, G., Tramontana, E. and Zappalà, G., 2014. A cloud-distributed GPU architecture for pattern identification in segmented detectors big-data surveys. *The Computer Journal*, *59*(3), pp.338-352.

Peng, H., Long, F., & Ding, C., 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on pattern analysis and machine intelligence, 27(8), pp. 1226-1238.

Rani, A., Foresti, G. L., &Micheloni, C., 2015. A neural tree for classification using convex objective function. Pattern Recognition Letters, pp.68, 41-47.

Reynès, C., Sabatier, R., Molinari, N. and Lehmann, S., 2008. A new genetic algorithm in proteomics: Feature selection for SELDI-TOF data. *Computational Statistics & Data Analysis*, *52*(9), pp.4380-4394.

Robnik-Šikonja, M. and Kononenko, I., 2003. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine learning*, *53*(1-2), pp.23-69.

Savvas, I.K., Sofianidou, G.N. and Kechadi, M.T., 2013. Applying the K-Means Algorithm in Big Raw Data Sets with Hadoop and MapReduce. *Big Data Management, Technologies, and Applications*, p.23.

Savvas, I.K. and Sofianidou, G.N., 2014, June. Parallelizing k-means algorithm for 1-d data using mpi. In *Wetice Conference (WETICE), 2014 IEEE 23rd International* (pp. 179-184). IEEE.

Savvas, I.K. and Sofianidou, G.N., 2016. A novel near-parallel version of k-means algorithm for n-dimensional data objects using mpi. *International Journal of Grid and Utility Computing*, *7*(2), pp.80-91.

Seewald, A.K. and Fürnkranz, J., 2001, September. An evaluation of grading classifiers. In *International symposium on intelligent data analysis* (pp. 115-124). Springer, Berlin, Heidelberg.

Sheikhpour, R., Sarram, M.A., Gharaghani, S. and Chahooki, M.A.Z., 2017. A survey on semi-supervised feature selection methods. *Pattern Recognition*, *64*, pp.141-158.

Shim, K., 2012. MapReduce algorithms for big data analysis. *Proceedings of the VLDB Endowment*, *5*(12), pp.2016-2017.

Song, G., Ye, Y., Zhang, H., Xu, X., Lau, R.Y. and Liu, F., 2016. Dynamic clustering forest: an ensemble framework to efficiently classify textual data stream with concept drift. *Information Sciences*, *357*, pp.125-143.

Song, Q., Ni, J. and Wang, G., 2013. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE transactions on knowledge and data engineering*, *25*(1), pp.1-14.

Stefanowski, J., 1998, September. The rough set based rule induction technique for classification problems. In In Proceedings of 6th European Conference on Intelligent Techniques and Soft Computing EUFIT (Vol. 98).

Tao, H., Ma, X.P. and Qiao, M.Y., 2013. Subspace Selective Ensemble Algorithm Based on Feature Clustering. *JCP*, *8*(2), pp.509-516.

Ting, K. M., & Witten, I. H., 1997. Stacking bagged and dagged models.

Tóth, R., Felici, F., Heuberger, P.S.C. and Van den Hof, P.M.J., 2008. Crucial aspects of zero-order hold LPV state-space system discretization. *IFAC Proceedings Volumes*, *41*(2), pp.4952-4957.

Vergara, J.R. and Estévez, P.A., 2014. A review of feature selection methods based on mutual information. *Neural computing and applications*, *24*(1), pp.175-186.

Ward, J.S. and Barker, A., 2013. Undefined by data: a survey of big data definitions. arXiv preprint arXiv:1309.5821.

Webb, G.I., 2000. Multiboosting: A technique for combining boosting and wagging. *Machine learning*, *40*(2), pp.159-196.

Xu, R.F. and Lee, S.J., 2015. Dimensionality reduction by feature clustering for regression problems. *Information Sciences*, *299*, pp.42-57.

Xue, B., Zhang, M. and Browne, W.N., 2013. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE transactions on cybernetics*, *43*(6), pp.1656-1671.
Xue, B., Zhang, M., Browne, W.N. and Yao, X., 2016. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, *20*(4), pp.606-626.
Xue, Y., Yin, X. and Jiang, X., 2016. Ensemble sufficient dimension folding methods for analyzing matrix-valued data. *Computational Statistics & Data Analysis*, *103*, pp.193-205.
Yijing, L., Haixiang, G., Xiao, L., Yanan, L. and Jinling, L., 2016. Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data. *Knowledge-Based Systems*, *94*, pp.88-104.
Zare, H., &Niazi, M., 2016. Relevant based structure learning for feature selection. Engineering Applications of Artificial Intelligence, 55, pp.93-102.

Appendix A: Dataset description

Number	Data	Number of Samples	Number of Feature	Number of Classes
1	IRIS	150	4	3
2	Ecoli	314	7	8
3	Glass	214	9	7
4	Breast	799	9	2
5	Wine	178	13	3
6	Heart	270	13	2
7	Letter	20000	16	26
8	Vehicle	946	18	4
9	Hepatitis	155	19	2
10	Segment	2310	19	7
11	WDBC	569	30	2
12	Ionosphere	350	34	2
13	Dermatology	336	35	7
14	Satellite	6435	36	7
15	Waveform	5000	40	3
16	Spambase	4601	57	2
17	Sonar	208	60	2
18	DNA	3186	60	3
19	Numerals	4000	100	10
20	Hill-Valley (Hill)	606	101	2
21	Arrhythmia	452	279	16
22	Lung	73	325	7
23	Madelon	4400	500	2
24	Colon	62	2000	2
25	Lymphoma	96	4026	9
26	Gisette	13500	5000	2
27	Nci9	60	9712	9
28	Arcene	900	10000	2
29	Cll_Sub_111 (CLL)	111	11340	3
30	Dexter	2600	20000	2
31	Dorothea	1950	100000	2
32	Gas sensor array under flow modulation (Gas 1)	58	120432	12
33	Pems-SF (Pems)	440	138672	7
34	Gas sensor array exposed to turbulent gas mixtures (Gas2)	180	150000	10
35	Twin gas sensor array (Twin gas)	640	480000	10
36	URL Reputation (URL)	2396130	3231961	2

Highlights (for review)

- Proposing a new ensemble method to classify high-dimensional datasets in a reasonable time
- Using HFC method to cluster features by maximizing accuracy and minimizing redundancy
- Using a modified GA in HFC method to improve the quality of clusters of features
- Developing a forest of neural trees with RBF nodes on the features of all of the clusters
- Transferring the knowledge of RBF synaptic weights to accelerate training of next RBF nodes
- Constructing a gating network to aggregate results of neural trees to improve accuracy