An Evolutionary Algorithm for Constructing a Decision Forest: Combining the Classification of Disjoints Decision Trees

Lior Rokach[1,*]
[1] *Department of Information System Engineering, Ben-Gurion University of the Negev, Be'er Sheva, Israel*

Decision forest is an ensemble classification method that combines multiple decision trees to in a manner that results in more accurate classifications. By combining multiple heterogeneous decision trees, decision forest is effective in mitigating noise that is often prevalent in real-world classification tasks. This paper presents a new genetic algorithm for constructing a decision forest. Each decision tree classifier is trained using a disjoint set of attributes. Moreover, we examine the effectiveness of using a Vapnik–Chervonenkis dimension bound for evaluating the fitness function of decision forest. The new algorithm was tested on various datasets. The obtained results have been compared to other methods, indicating the superiority of the proposed algorithm. © 2008 Wiley Periodicals, Inc.

1. INTRODUCTION AND MOTIVATION

Data mining is the science, art, and technology of exploring large and complex bodies of data to discover useful patterns. The accessibility and abundance of information today makes data mining a matter of considerable importance and necessity. Theoreticians and practitioners are continually seeking improved techniques to make the process more efficient, cost-effective, and accurate.

One of the most practical techniques used in data mining is classification. The aim of classification is to build a classifier (also known as a classification model) by induction from a set of preclassified instances. The classifier can then be used for classifying unlabeled instances.

One of the most promising and popular approaches are decision trees.[1] Decision trees are simple yet successful techniques for predicting and explaining the relationship between some measurements about an item and its target value. In addition to their use in data mining, decision trees, which originally derived from logic, management, and statistics, are today highly effective tools in other areas such as text mining.

*Author to whom all correspondence should be addressed: e-mail: liorrk@bgu.ac.il.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 23, 455–482 (2008)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20277

Decision forest is a pattern recognition method, which combines the results of multiple distinct but comparable decision tree classifiers to reach a consensus prediction.[2] In fact, ensemble methodology imitates our second nature to seek several opinions before making any crucial decision. We weigh the individual opinions, and combine them to reach a final decision. Improvement of accuracy and stability is observed in experiments and applications, especially in life science (see, for instance, Tong et al.[3]; Hong et al.[4]; Xie et al.[5]). In this paper, we are interested in constructing a decision forest for high-dimensional datasets.

It is well known that the required number of labeled samples for supervised classification increases as a function of dimensionality.[6] Fukunaga[7] showed that the required number of training samples is linearly related to the dimensionality for a linear classifier and to the square of the dimensionality for a quadratic classifier. In terms of nonparametric classifiers like decision trees, the situation is even more severe. It has been estimated that, as the number of dimensions increases, the sample size needs to increase exponentially to have an effective estimate of multivariate densities.[8]

Bellman[9] was the first to define this phenomenon as the "curse of dimensionality," while working on complicated signal processing problems. Techniques like decision trees inducers that are efficient in low dimensions fail to provide meaningful results when the number of dimensions increases beyond a "modest" size. Furthermore, smaller classifiers, involving fewer attributes (probably less than 10), are much more understandable by humans. Smaller classifiers are also more appropriate for user-driven data mining techniques such as visualization.

Most methods for dealing with high dimensionality focus on attribute selection techniques, i.e., selecting a single subset of attributes upon which the inducer (induction algorithm) will run, while ignoring the rest. The selection of the subset can be done manually using prior knowledge to identify irrelevant feature or attribute selection algorithms. In the last decade, many researchers have shown increased interest in attribute selection. Consequently many attribute selection algorithms have been proposed, with some demonstrating remarkable improvements in accuracy. Since the subject is too wide to survey here, the reader is referred to Liu and Motoda[10] for further reading.

Despite the popularity of attribute selection methodologies, there are several drawbacks in using them in overcoming the "curse of dimensionality":

- The assumption that a large set of input attributes can be reduced to a small subset of relevant attributes is not always true; in some cases, the target feature is actually affected by most of the input attributes and removing features will cause a significant loss of important information.
- The outcome (i.e., the subset) of many algorithms for attribute selection (e.g., almost any of the algorithms that are based upon the wrapper methodology) is strongly dependent on the training set size. That is, if the training set is small, the size of the reduced subset will also be small. Consequently, relevant features might be lost. Accordingly, the induced classifiers might achieve lower accuracy compared to classifiers that have access to all relevant features.
- In some cases, even after eliminating a set of irrelevant features, the researcher is left with a relatively large number of relevant features.

- The backward elimination strategy, used by some methods, is extremely inefficient for working with large-scale databases, where the number of original features is more than 100.

Several researchers have shown that creating an ensemble from a set of reducts can be appropriate for classification tasks with large number of features (see, for instance, Kusiak,[11] Maimon and Rokach[12]). In our previous paper,[13] we have examined the idea of attribute reducts for generalizing the task of attribute selection. Attribute selection aims to provide a single representative set of features from which a classifier is constructed. On the other hand, attribute reducts decomposes the original set of features into several subsets, and builds a classifier for each subset. Thus, a set of classifiers are trained such that each classifier employs a different subset of the original features set. Subsequently, an unlabeled instance is classified by combining the classifications of all classifiers. This method potentially facilitates the creation of a classifier for high-dimensionality datasets without the above-mentioned drawbacks of attribute selection.

As part of our previous work, a simple hill-climbing algorithm called DOG (decomposed-oblivious-gain) has been proposed. This algorithm searches for the quasi-optimal disjoints attribute reducts using incremental oblivious decision trees. One limitation with the DOG algorithm is that it has no backtracking capabilities (for instance, removing a single feature from a subset or removing an entire subset). Furthermore, the search of DOG begins from an empty partitioning structure, which may lead to subsets with relatively small number of features. This paper suggests a more profound exploration of the search space. Because performing exhaustive search is intractable for large problems, we decided to employ genetic algorithm.

Evolutionary algorithms (EAs) are stochastic search algorithms inspired by the process of Darwinian evolution. The motivation for applying EAs to data mining tasks is that they are robust, adaptive search techniques that perform a global search in the solution space.[14] When EA is well designed, it continually consider new solutions. Thus, it can be viewed as an "anytime" learning algorithm.[15] Such a learning algorithm should produce a good-enough solution quite quickly, then continue to search the solution space, reporting the new "best" solution whenever one is found. This is important since, there are, in which the decision maker is willing to wait for weeks, or even months, if a learning system can produce an improved solutions.

Genetic algorithm is a popular type of EA and was successfully used for attribute selection. Empirical comparisons between GAs and other kinds of attribute selection methods can be found.[16,17] In general, these empirical comparisons show that GAs, with their associated global search in the solution space, usually (though not always) obtain better results than local search-based attribute selection methods. In particular, Kudo and Skalansky[17] compared a GA with 14 nonevolutionary attribute selection methods (some of them variants of each other) across eight different datasets. The authors concluded that the advantages of the global search associated with GAs over the local search associated with other algorithms is particularly important in datasets with a "large" number of attributes, where "large" was considered over 50 attributes in the context of their datasets. Hsu[18] has developed the idea of using genetic algorithms for attribute selection. Specifically, he developed two

genetic algorithm wrappers: one for the variable selection problem for decision tree inducers and one for the variable ordering problem for Bayesian network structure learning.

AdaBoost[19] is a popular implementation of the ensemble methodology, which sequentially constructs a series of classifiers, where the training instances that are wrongly classified by a certain classifier will get a higher weight in the training of its subsequent classifier. The classifiers predictions are combined via weighted voting where the weights are determined by the algorithm itself based on the training error of each classifier.

There are several works in the literature that propose to construct an ensemble of classifiers by manipulating the input attribute set for creating the ensemble member. The idea is to simply give each classifier a different projection of the training set. Recently, Hu et al.[20] propose the EROS algorithm that generates a set of reducts, and then each reduct is used to train a base classifier.

Chawla et al.[21] claim that classic ensemble techniques have limitations on massive datasets, because the size of the dataset can become a bottleneck. Therefore, Chawla et al.[21] suggest use of disjoints subsets. They argue that this approach will not only overcome the issue of exceeding memory size but will also lead to creating a set of diverse and accurate classifiers, each built from a disjoint subset, but the aggregate processing all of the data. This can result in an improvement in performance that might not be possible by subsampling. Similarly Rokach and Maimon (Ref. 40) use feature set decomposition to analyze the effect of manufacturing setting on the product's quality.

Opitz and Shavlik[22] applied GAs to ensembles. However, its genetic operators were designed explicitly for hidden nodes in knowledge-based neural networks and the algorithm does not work well with problems lacking prior knowledge. In a later research, Opitz[16] uses genetic search for ensemble attribute selection. This genetic ensemble attribute selection strategy begins with creating an initial population of classifiers where each classifier is generated by randomly selecting a different subset of features. Then, new candidate classifiers are continually produced by using the genetic operators of crossover and mutation on the feature subsets. The final ensemble is composed of the most fitted classifiers.

In this paper, we are interested in decomposing the original attribute set into several disjoints subsets. Disjoints attribute reduction is a generalization of the attribute selection task. Moreover, it is as specific case of ensemble methodology in which members are using disjoints attribute reducts. Given the positive evidence of using genetic algorithm for attribute selection tasks from one hand and for creating ensemble of classifiers from the other hand, using genetic algorithm in this case is, therefore, self-evident. In fact a similar idea has been proposed as a part of position paper.[23] However, there have been no reports about whether this idea was implemented and whether it can improve classification performance.

All the above-mentioned genetic algorithms for attribute selection have used the wrapper approach for evaluating the fitness function. In this approach, a certain solution is evaluated by repeatedly sampling the training set and measuring the accuracy of the inducers obtained for feature subsets over a holdout validation dataset. The main advantages of this approach are the fact that it generates reliable

evolutions and that it can be used for any induction algorithm. Nevertheless the fact that the wrapper procedure repeatedly executes the inducer, is considered major drawback. For this reason, wrappers may not scale well to large datasets containing many features.

The aim of this work is to examine whether genetic algorithm-based disjoints attribute reducts can improve the classification performance. For this purpose, we are using a specific encoding schema. Using theoretical results, it is has been shown that this encoding is more suitable than more straightforward encoding schemas.[24] Moreover, to avoid the above mentioned drawbacks of the wrapper approach, a Vapnik–Chervonenkis dimension bound for decision forest is used to evaluate the fitness function. A caching mechanism is used to additionally reduce the computational complexity of the genetic algorithm. The superiority of the suggested algorithm over other methods is illustrated on various datasets.

2. PROBLEM FORMULATION

In a typical classification problem, a training set of labeled examples is given and the goal is to form a description that can be used to predict previously unseen examples. The training set can be described in a variety of languages, most frequently, as a collection of records that may contain duplicates. Each record is described by a vector of attribute values. The notation A denotes the set of input attributes containing n attributes: $A = \{a_1, \ldots, a_i, \ldots, a_n\}$ and y represents the class variable or the target attribute. Attributes (sometimes referred to as fields, variables, or features) are typically one of two types: categorical (values are members of a given set) or numeric (values are real numbers). When the attribute a_i is categorical, it is useful to denote its domain values by $dom(a_i) = \{v_{i,1}, v_{i,2}, \ldots, v_{i,|dom(a_i)|}\}$, where $|dom(a_i)|$ stands for its finite cardinality. In a similar way, $dom(y) = \{c_1, \ldots, c_{|dom(y)|}\}$ represents the domain of the target attribute. Numeric attributes have infinite cardinalities.

The instance space (the set of all possible examples) is defined as a Cartesian product of all the input attributes domains: $X = dom(a_1) \times dom(a_2) \times \cdots \times dom(a_n)$. The universal instance space (or the labeled instance space) U is defined as a Cartesian product of all input attribute domains and the target attribute domain, i.e., $U = X \times dom(y)$. The training set consists of a set of m records and is denoted as $S = (\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_m, y_m \rangle)$ where $\boldsymbol{x}_q \in X$ and $y_q \in dom(y)$.

Usually, it is assumed that the training set records are generated randomly and independently according to some fixed and unknown joint probability distribution D over U. Note that this is a generalization of the deterministic case when a supervisor classifies a record using a function $y = f(x)$.

The notation DTI represents a decision tree inducer, and $DTI(S)$ represent a decision tree which was induced by activating the inducer DTI onto dataset S. In this case it is possible to estimate the conditional probability $\hat{P}_{\mathrm{DTI}(S)}(y = c_j | a_i = x_{q,i}\ i = 1, \ldots, n)$ of an observation x_q. Note the addition of the "hat" —^—to the conditional probability estimation is used for distinguishing it form the actual conditional probability.

Figure 1 presents the structure of the proposed system. The gray boxes depict various components in the decomposition system, whereas the transparent boxes

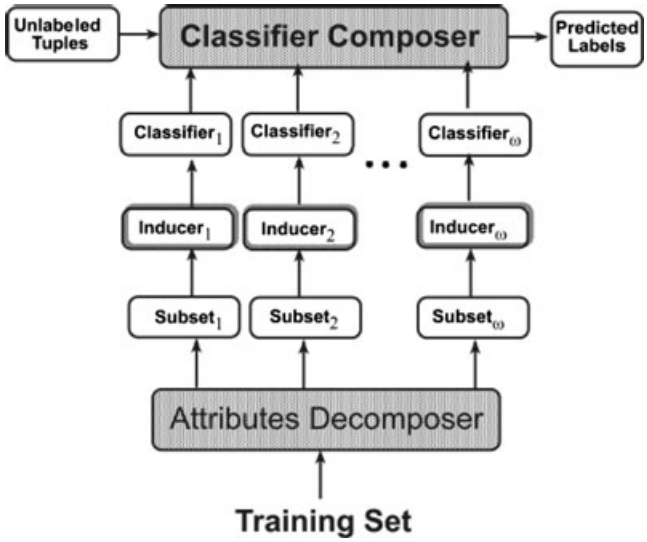

Figure 1. The structure of the proposed framework.

are considered as input/output. There is some work to do for constructing such an ensemble system. First, an algorithm for searching multiple disjoints reducts should be proposed. Second, one should select learning algorithms to train base classifiers with the resulting reducts. In this general framework, a different inducer can be used for each subset. Nevertheless, one may use the same inducer for each subset and still obtain different classifiers. Finally, a strategy of combining the base classifiers is required for constructing an efficient and powerful ensemble system.

In this paper we focus on the first task, i.e., finding the best partitioning of the input attribute set, such that if a specific decision tree inducer is run on each attribute reduct, then the combination of the generated decision trees will have the highest possible accuracy. Consequently, the problem can be formally phrased as follows:

> Given an inducer DTI, a combination method C, and a training set S with input feature set $A = \{a_1, a_2, \ldots, a_n\}$ and target feature y from a distribution D over the labeled instance space, the goal is to find an optimal partitioning $Z_{\text{opt}} = \{G_1, \ldots G_k \ldots, G_\omega\}$ of the input feature set A into ω mutually exclusive subsets $G_k = \{a_{\alpha_k(j)} \,|\, j = 1, \ldots, l_k\}$; $k = 1, \ldots, \omega$ that are not necessarily exhaustive such that the generalization error of the induced decision trees $\text{DTI}(\pi_{G_k \cup y} S)$; $k = 1, \ldots, \omega$ combined using method C, will be minimized over the distribution D.

where $G_k = \{a_{\alpha_k(j)} | j = 1, \ldots, l_k\}$ indicates the kth subset that contains l_k input attributes such that $\alpha_k : \{1, \ldots, l_k\} \rightarrow \{1, \ldots, n\}$ is a function that maps the attribute index j to the original attribute index in the set A. $R_k = \{i | \exists j \in \{1, \ldots, l_k\}$ s.t. $\alpha_k(j) = i\}$ denotes the corresponding indexes of subset k in the complete attribute set A and $\pi_{G_k \cup y} S$ represents the corresponding projection of S.

This paper focuses on feature set partitioning designed for decision trees, which are combined using the Naive Bayes combination, namely C is the Naive Bayes

combination. In the Naive Bayes combination, a classification of a new instance is based on the product of the conditional probability of the target feature, given the values of the input attributes in each subset. Mathematically, it can be formulated as follows:

$$v_{\mathrm{MAP}}(x_q) = \underset{\substack{c_j \in dom(y) \\ \hat{P}_{\mathrm{DTI}(S)}(y=c_j)>0}}{\arg\max} \hat{P}_{\mathrm{DTI}(S)}(y = c_j) \cdot \prod_{k=1}^{\omega} \frac{\hat{P}_{\mathrm{DTI}(\pi_{G_k \cup y} S)}(y = c_j | a_i = x_{q,i}\ i \in R_k)}{\hat{P}_{\mathrm{DTI}(S)}(y = c_j)} \tag{1}$$

or

$$v_{\mathrm{MAP}}(x_q) = \underset{\substack{c_j \in dom(y) \\ \hat{P}_{DTI(S)}(y=c_j)>0}}{\arg\max} -(\omega - 1) \log\left(\hat{P}_{\mathrm{DTI}(S)}(y = c_j)\right) + \sum_{k=1}^{\omega} \log\left(\hat{P}_{\mathrm{DTI}(\pi_{G_k \cup y} S)}(y = c_j | a_i = x_{q,i}\ i \in R_k)\right) \tag{2}$$

Recall that R_k denotes the correspondence indexes of subset k in the complete feature set A. In case of decision trees, $\hat{P}_{\mathrm{DTI}(\pi_{G_k \cup y} S)}(y = c_j | a_i = x_{q,i}\ i \in R_k)$ can be estimated by using the appropriate frequencies in the relevant leaf. However, using the frequency as is will typically overestimate the probability. To avoid this phenomenon, it is useful to perform the Laplace correction.[25] According to Laplace's law of succession, the probability of the event $y = c_i$ where y is a random variable and c_i is possible outcome of y which has been observed mi times out of m observations is: $(m_i + kp_{\text{a-priori}})/(m + k)$, where $p_{\text{a-priori}}$ is an a-priori probability estimation of the event and k is the *equivalent sample size* that determines the weight of the a-priori estimation relative to the observed data.

It should be noted that the optimal partitioning structure is not necessarily unique. Furthermore, it is not obligatory that all input attributes actually belong to one of the subsets. Consequently, the problem can be treated as an extension of the attribute selection problem, i.e., finding the optimal partitioning of the form $Z_{\text{opt}} = \{G_1\}$, as the nonrelevant features are in fact $NR = A - G_1$. Moreover, when using a Naive Bayes for combining the classifiers as in this case, the Naive Bayes method can be treated as specific partitioning: $Z = \{G_1, G_2, \ldots, G_n\}$, where $G_i = \{A_i\}$.

DEFINITION 1. *(*Classification-preservation partitioning). *The partitioning* $Z = \{G_1, \ldots, G_k, \ldots, G_\omega\}$ *is said to be classification-preservation if for each instance in the support of* $P(x_q)$*, the following is satisfied:*

$$\underset{\substack{c_j \in dom(y) \\ \hat{P}(y=c_j)>0}}{\arg\max} \frac{\prod_{k=1}^{\omega} P(y = c_j \,|a_i = x_{q,i}\ i \in R_k)}{P(y = c_j)^{\omega-1}}$$

$$= \underset{\substack{c_j \in dom(y) \\ P(y=c_j)>0}}{\arg\max} P(y = c_j \,|a_i = x_{q,i} \; i = 1, \ldots, n) \tag{3}$$

Since Duda and Hart[26] showed that the right term of the equation is optimal, it follows that classification-preservation partitioning is also optimal. The importance of finding classification-preservation partitioning is derived from the fact that in real problems with limited training sets, it is easier to approximate probabilities with fewer dimensions.

The following four Lemmas are presented to shed light on the suggested problem. This set of lemmas defines *classification preservation* and demonstrates that conditional independence is not a necessary precondition thereof. More specifically, these lemmas show that the Naive Bayes combination can be useful in various cases of separable functions even when the Naive assumption of conditional independence is not necessarily fulfilled. Furthermore, because these lemmas provide the optimal partitioning structures they can be used for evaluating the performance of the algorithms proposed in Section 3. The proofs of these lemmas are straightforward and are provided in Rokach.[25]

LEMMA 1. (Sufficient condition). *Let Z be a partitioning that satisfies the following conditions:*

1. The subsets $G_k, k = 1, \ldots, \omega$ and the $NR = A - \bigcup_{k=1}^{\omega} G_k$ are conditionally independent given the target feature;
2. The NR set and the target feature are independent.

then Z is classification-preservation.

Lemma 1 represents a sufficient condition for classification-preservation. It is important to note that it does not represent a necessary condition; as illustrated in the following lemma:

LEMMA 2. (The additive case). *Let* $A = \{a_1, \ldots, a_l, \ldots, a_n\}$ *be a group of n independent input binary features and let* $Z = \{G_1, \ldots, G_\omega\}$ *be a partitioning, then if the target feature follows the function:*

$$y = 2^0 \cdot f_1(a_i, i \in R_1) + 2^1 \cdot f_2(a_i, i \in R_2) + \cdots + 2^{\omega-1} f_\omega(a_i, i \in R_\omega)$$

where $f_1, \ldots, f_\omega$ *are Boolean functions and* $R_1, \ldots, R_\omega$ *are mutually exclusive then Z is classification-preservation.*

LEMMA 3. (The read-once DNF case). *Let* $A = \{a_1, \ldots, a_l, \ldots, a_n\}$ *denote a group of n independent input binary features and let* $Z = \{G_1, \ldots, G_\omega\}$ *denotes a partitioning, then if the target feature follows the function:*

$$y = f_1(a_i, i \in R_1) \vee f_2(a_i, i \in R_2) \vee \ldots \vee f_\omega(a_i, i \in R_\omega)$$

or

$$y = f_1(a_i, i \in R_1) \wedge f_2(a_i, i \in R_2) \wedge \ldots \wedge f_\omega(a_i, i \in R_\omega)$$

where $f_1, \ldots, f_\omega$ are Boolean functions and $R_1, \ldots, R_\omega$ are mutually exclusive, then Z is classification-preservation.

Lemmas 2 and 3 illustrate that although the conditionally independence requirement is not fulfilled it is still possible to find a classification-preservation partitioning.

LEMMA 4. (The XOR case). *Let $A = \{a_1, \ldots, a_i, \ldots, a_n\}$ be a group of n input binary features distributed uniformly. If the target feature behaves as $y = a_1 \oplus a_2 \oplus \ldots \oplus a_n$ then there is no partitioning beside $Z = \{A\}$, which is classification-preservation.*

Lemma 4 shows that there are problems such that no classification-preservation partitioning can be found, beside the obvious one.

The number of combinations that n^* input attributes may be decomposed into exactly ω relevant subsets is

$$P(n^*, \omega) = \frac{1}{\omega!} \sum_{j=0}^{\omega} \binom{\omega}{j} (-1)^j (\omega - j)^{n^*} \tag{4}$$

Evidently the number of combinations that n^* input attributes may be decomposed into up to n^* subsets is

$$C(n^*) = \sum_{\omega=1}^{n^*} P(n^*, \omega) = \sum_{\omega=1}^{n^*} \frac{1}{\omega!} \sum_{j=0}^{\omega} \binom{\omega}{j} (-1)^j (\omega - j)^{n^*} \tag{5}$$

Because the feature set partitioning problem defined above, it is possible that part of the input feature will not be used by the inducers (the irrelevant set), then the total search space is

$$T(n) = \sum_{n^*=0}^{n} \binom{n}{n^*} C(n^*) = \sum_{n^*=0}^{n} \binom{n}{n^*} \sum_{\omega=1}^{n^*} \frac{1}{\omega!} \sum_{j=0}^{\omega} \binom{\omega}{j} (-1)^j (\omega - j)^{n^*} \tag{6}$$

Equation 6 implies that exhaustive search is intractable for large problems. Thus, heuristic search algorithm is required. The next section presents a genetic algorithm for solving this problem.

3. A GENETIC ALGORITHM FOR CONSTRUCTING A DECISION FOREST

3.1. Overview

To solve the problem defined in Section 2, we suggest using a genetic algorithm search procedure as proposed by Freitas.[15]

Genetic algorithms begin by randomly generating a population of L candidate solutions. Given such a population, a genetic algorithm generates a new candidate solution (population element) by selecting two of the candidate solutions as the parent solutions. This process is termed reproduction. Generally, parents are selected randomly from the population with a bias toward the better candidate solutions. Given two parents, one or more new solutions are generated by taking some characteristics of the solution from the first parent (the "father") and some from the second parent (the "mother"). This process is termed crossover. For example, in genetic algorithms that use binary encoding of n bits to represent each possible solution, we might randomly select a crossover bit location denoted as o. Two descendants' solutions could then be generated. The first descendant would inherit the first o string characteristics from the father and the remaining $n - o$ characteristics from the mother. The second descendant would inherit the first o string characteristics from the mother and the remaining $n - o$ characteristics from the father. This is type of crossover is the most common, and it is termed one-point crossover. Crossover is not necessarily applied to all pairs of individuals selected for mating: a $P_{\text{crossover}}$ probability is used to decide whether crossover will be applied. If crossover is not applied, the offspring are simply duplications of the parents.

Finally, once descendant solutions are generated, genetic algorithms allow characteristics of the solutions to be changed randomly in a process know as mutation. In the binary encoding representation, according to a certain probability (P_{mut}) each bit is changed from its current value to the opposite value. Once a new population has been generated, it is decoded and evaluated. The process continues until some termination criterion is satisfied.

```
Create initial population of individuals
(candidate solutions)
Compute the fitness of each individual
REPEAT
Select individuals based on fitness
Apply genetic operators to selected individuals,
creating new individuals
Compute fitness of each of the new individuals
Update the current population
(new individuals replace old individuals)
UNTIL (stopping criteria)
```

A Pseudocode for GA

To implement a genetic algorithm one is required to provide a schema for encoding, manipulating, and evaluating the solution. The following sections present the schemes suitable to the problem discussed in this paper.

3.2. Encoding and Genetic Operators

A candidate solution consists mainly of values of variables—in essence, data. In particular, GA individuals are usually represented by a fixed-length linear genome.

A straightforward individual representation for feature set partitioning consists simply of a string of n integers. Recall that n is the number of features. The ith integer, $i = 1, \ldots, n$, can take the value $0, \ldots, n$, indicating to which subset (if any) the ith attribute belongs. A value of 0 indicates that the corresponded attribute is not selected and filtered out. For instance, in a 10-attribute dataset, the individual "2 1 0 3 2 2 1 0 3 2" represents a candidate solution where the 2nd and the 7th attributes are located in the first set. The 1st, the 5th, the 6th, and the 10th are located in the second set. The 4th and the 9th are located in the third set. All other attributes are filtered out. This individual representation is simple, and traditional one-point crossover operator can be easily applied. As to the mutation operator, according to a certain probability (P_{mut}) each integer is changed from its current value to a different valid value.

The last representation has redundancy, i.e., the same solution can be represented in several ways. For instance the illustrated solution "2 1 0 3 2 2 1 0 3 2" can be also represented as "3 2 0 1 3 3 2 0 1 3." Moreover, similar solutions can be represented in quite different ways. This property can lead to situations in which the offspring are dissimilar to their parents. Besides being not compact, the above encoding may result in a slow convergence of the genetic algorithm. A GA converges when most of the population is identical, or in other words, the diversity is minimal.[27] Louis and Rawlins[27] analyzed the convergence of binary strings using the Hamming distance and showed that traditional crossover operators (such as one-point crossover operator) does not change the average Hamming distance of a given population. In fact selection is responsible to the Hamming distance convergence. Therefore, we should look for encoding with similar properties.

We begin by defining a measure called partitioning structural distance. This measure can be used to determine the distance of two partitioning structures as following:

DEFINITION 2. Partitioning structural distance (DSD)

$$\delta\left(Z^1, Z^2\right) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2 \cdot \eta\left(a_i, a_j, Z^1, Z^2\right)}{n \cdot (n-1)} \tag{7}$$

where $\eta(a_i, a_j, Z^1, Z^2)$ is a binary function that returns the value "0" if the features a_i, a_j belong to the same subset in both partitioning structures Z^1, Z^2 or if a_i, a_j belong to different subsets in both partitioning structures. In all other cases, the function returns the value "1".

For example, given that $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$, $Z^1 = \{\{a_4, a_2\}; \{a_5, a_3\}\}$, and $Z^2 = \{\{a_1, a_3, a_5\}; \{a_2, a_4\}\}$ then:

$$\delta(Z^1, Z^2) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2 \cdot \eta(a_i, a_j, Z^1, Z^2)}{n \cdot (n-1)}$$

$$= \frac{2}{5 \cdot 6}(\eta(a_1, a_2, Z^1, Z^2) + \eta(a_1, a_3, Z^1, Z^2)$$

$$+\eta(a_1, a_4, Z^1, Z^2) + \eta(a_1, a_5, Z^1, Z^2) + \eta(a_1, a_6, Z^1, Z^2) + \eta(a_2, a_3, Z^1, Z^2)$$

$$+\eta(a_2, a_4, Z^1, Z^2) + \eta(a_2, a_5, Z^1, Z^2) + \eta(a_2, a_6, Z^1, Z^2) + \eta(a_3, a_4, Z^1, Z^2)$$

$$+\eta(a_3, a_5, Z^1, Z^2) + \eta(a_3, a_6, Z^1, Z^2) + \eta(a_4, a_5, Z^1, Z^2) + \eta(a_4, a_6, Z^1, Z^2)$$

$$+\eta(a_5, a_6, Z^1, Z^2))$$

$$= \frac{2}{30}(1+1+1+1+1+0+0+0+0+0+0+0+0+0+0) = \frac{5}{15}$$

It is important to note that the structural distance measure is an extension of the Rand index (Rand 1971[39]) developed for evaluating clustering methods.

By using an adjacency matrix-like encoding, one can represent any partitioning structure as $n \times n$ matrix in which cell (i, j) gets the value "1" if attributes a_i and a_j are located in the same group, cell (i, j) gets the value "–1" if attributes a_i and a_j are both filtered out and it gets the value "0" otherwise. The values on the diagonal indicate whether each attribute filtered out (–1) or not (1). For example, Table I illustrates the representation of $Z^1 = \{\{a_4, a_2\}; \{a_5, a_3\}\}$ given that $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$. Note that because the above matrix is always symmetric, we can specify only the upper triangle.

Table I. Illustration of adjacency matrix-like encoding.

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	−1	0	0	0	0	−1
a_2	0	1	0	1	0	0
a_3	0	0	1	0	1	0
a_4	0	1	0	1	0	0
a_5	0	0	1	0	1	0
a_6	−1	0	0	0	0	−1

DEFINITION 3. Encoding matrix A is said to be well defined if

1. *Commutative:* $\forall i \neq j; cell\,(i, j) = cell\,(j, i)$
2. *Transitive:* $\forall i \neq j \neq k; if\ cell\,(i, j) \neq 0 and\ cell\,(i, k) \neq 0\ then\ cell\,(j, k) \neq 0$
3. *Sign Property:* $\forall i \neq j; if\ cell\,(i, j) \neq 0\ then\ cell\,(i, j) = cell\,(i, i)$

We now suggest a new crossover operator called "groupwise crossover" (GWC) that works in the following way. We select one anchor subset from the subsets that define the first parent partitioning and one anchor subset from the subsets that define the second parent partitioning (the selected subset can be also the filtered out subset). The anchor subsets are used as-is without any addition or diminution of attributes.

The first offspring is created by copying the columns and rows of the attributes that belong to the first selected anchor subset from the first parent. All remaining cells are filled in with the corresponding values that are obtained from the second parent. The second offspring is similarly created using the second anchor subset by copying the appropriate columns and the rows from the second parent and then fill in the remaining cells with the corresponding values from the first parent.

Example. Assuming that two partitioning structures $Z^1 = \{\{a_4, a_2\}; \{a_5, a_3\}\}$ and $Z^2 = \{\{a_2, a_6\}; \{a_1, a_4, a_3\}\{a_5\}\}$ are given over the feature set $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$. In order to perform GWC operator, two anchor subsets are selected, one from each partitioning: $\{a_2, a_4\}$from Z^1 and $\{a_1, a_4, a_3\}$from Z^2. Table II illustrates representations of the Z^1 and Z^2 and their offspring Z^3 and Z^4. Z^3 is obtained by keeping the group $\{a_2, a_4\}$ and the remaining cells are copied from Z^2. Z^4 is obtained by keeping the group $\{a_1, a_4, a_3\}$ and the remaining cells are copied from Z^1. Thus, $Z^3 = \{\{a_2, a_4\}; \{a_1, a_3\}; \{a_5\}; \{a_6\}\}$ and $Z^4 = \{\{a_1, a_4, a_3\}; \{a_5\}; \{a_2\}\}$. The highlighted cells indicate the selected group that was copied into the offspring.

The following set of lemmas defines *well definition* for an adjacency matrix representation of an attribute partition and demonstrates preservation of well definition under a crossover operator (GWC) defined over attribute partitions;

LEMMA 5. (Structural distance measure properties). *The structural distance measure has the following properties:*

1. *Symmetry*: $\delta(Z^1, Z^2) = \delta(Z^2, Z^1)$
2. *Positivity:* $\delta(Z^1, Z^2) = 0$ IIf $Z^1 = Z^2$
3. *Triangular Inequality*: $\delta(Z^1, Z^2) \leq \delta(Z^1, Z^3) + \delta(Z^2, Z^3)$

LEMMA 6. *A projection of well-defined encoding matrix is well-defined encoding matrix.*

LEMMA 7. *Performing GWC operator on two well-defined encoding matrices generates a new well-defined encoding matrix.*

Table II. Illustration of GWC operator.

Z^1

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	−1	0	0	0	0	−1
a_2	0	1	0	1	0	0
a_3	0	0	1	0	1	0
a_4	0	1	0	1	0	0
a_5	0	0	1	0	1	0

Z^2

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	1	0	1	1	0	0
a_2	0	1	0	0	0	1
a_3	1	0	1	1	0	0
a_4	1	0	1	1	0	0
a_5	0	0	0	0	1	0

Z^3

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	1	0	1	0	0	0
a_2	0	1	0	1	0	0
a_3	1	0	1	0	0	0
a_4	0	1	0	1	0	0
a_5	0	0	0	0	1	0
a_6	0	0	0	0	0	1

Z^4

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	1	0	1	1	0	0
a_2	0	1	0	0	0	0
a_3	1	0	1	1	0	0
a_4	1	0	1	1	0	0
a_5	0	0	0	0	1	0
a_6	0	0	0	0	0	−1

LEMMA 8. *Operator GWC creates offspring that their distance is not greater than the distance of their parents.*

Lemma 8 indicates that the GWC operator together with the proposed encoding does not slow down the convergence of the genetic algorithm. Together with the selection process that prefers solutions with higher fitness values, one can ensure that the algorithm does converge.[27]

As to the mutation operator, according to a certain probability (P_{mut}) each attribute can be cut off from its original group and join another randomly selected group.

3.3. Fitness Function

In each iteration, we have to create a new population from the current generation. The selection operation determines which parent chromosomes participate in producing offspring for the next generation. Usually, members are selected for mating with a selection probability proportional to their fitness values. The most

common way to implement this method is to set the selection probability p_i equal to

$$p_i = \frac{f_i}{\sum_j f_j} \tag{8}$$

For a classification problem, the fitness value f_i of the ith member can the complement to 1 of the *generalization error*. Note that using training error as-is is not sufficient for evaluating classifiers due to overfitting phenomena.

The most straightforward to estimate generalization error is to use the wrapper procedure. In this approach, the partitioning structure is evaluated by repeatedly sampling the training set and measuring the accuracy of the inducers obtained for this partitioning on an unused portion of the training set. This is the most common approach for evaluate the fitness function in attribute selections problems. However as stated in Section 1, the fact that the wrapper procedure repeatedly executes the inducer, is considered major drawback.

An alternative approach for evaluating performance is to use the generalization error bound in terms of the training error and concept size. In this paper, we decided to use the VC theory for evaluating the generalization error bound. According to the VC theory the bound on the generalization error of hypothesis space H with finite VC-dimension d is given by:

$$|\varepsilon(h, D) - \hat{\varepsilon}(h, S)| \leq \sqrt{\frac{d \cdot \left(\log \frac{2m}{d} + 1\right) - \log \frac{\delta}{4}}{m}} \quad \forall h \in H \quad \forall \delta > 0 \tag{9}$$

with probability of $1 - \delta$ where $\hat{\varepsilon}(h, S)$ represents the training error of classifier h measured on training set S of cardinality m and $\varepsilon(h, D)$ represents the generalization error of the classifier h over the distribution D.

To use the bound (Equation 9), one needs to measure the VC-dimension. The VC dimension for a set of indicator functions is defined as the maximum number of data points that can be shattered by the set of admissible functions. By definition, a set of m points is shattered by a concept class if there are concepts (functions) in the class that split the points into two classes in all of the 2^m possible ways. The VC dimension might be difficult to compute accurately and it depends on the induction algorithm.

In this paper, concentrates on decision tree inducers and more specifically on oblivious decision trees. Oblivious decision trees are found to be effective for attribute selection, which is a simplified case of the problem solved here. Oblivious decision trees are decision trees, for which all nodes at the same level test the same feature.

Figure 2 illustrates a typical oblivious decision tree with four input attributes: Glucose level (G), Age (A), Hypertension (H) and Pregnant (P), and the Boolean target feature representing whether that patient suffers from diabetes. Each layer is uniquely associated with an input feature by representing the interaction of that

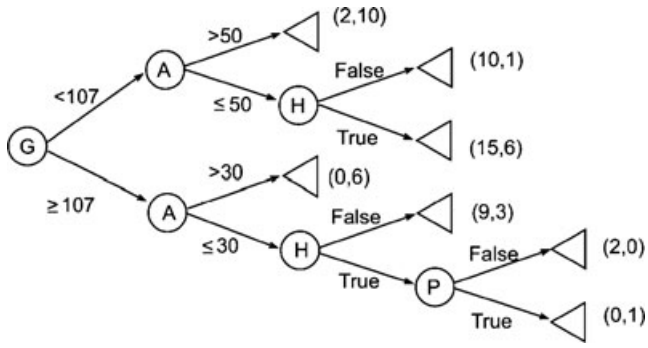

Figure 2. Illustration of oblivious decision tree.

feature and the input attributes of the previous layers. The number that appears in the terminal nodes indicates the number of instances that fit this path. For example, regarding patients whose glucose level is less than 107 and whose age is greater than 50, ten are positively diagnosed with diabetes while two are not diagnosed with diabetes.

The principal difference between the oblivious decision tree and a regular decision tree structure is the constant ordering of input attributes at every terminal node of the oblivious decision tree, the property which is necessary for minimizing the overall subset of input attributes (resulting in dimensionality reduction). Therefore despite its restriction, an oblivious decision tree is found to be effective as a attribute selection procedure. Almuallim and Dietterich,[28] as well as Schlimmer,[29] have proposed forward attribute selection procedure by constructing oblivious decision trees while Langley and Sage[30] suggested backward selection using the same means.

In the case of feature set partitioning, each subset is represented by an oblivious decision tree and each feature is located on a different layer. As a result, adding a new feature to a subset is performed by adding a new layer and connecting it to the nodes of the last layer. The nodes of a new layer are defined as the Cartesian product combinations of the previous layer's nodes with the values of the new added feature. To avoid unnecessary splitting, the algorithm splits a node only if it is useful. In this paper we split a node, if the information gain of the new feature in this node is strictly positive.

The unique structure of oblivious decision tree is very convenient to our genetic algorithm approach. Moving from one generation to the other usually require a small changes on the subset structures. Because each feature is located on a different layer, it relatively easy to add or remove features incrementally; as opposed to regular decision tree inducers in which each iteration of the search may require generating the decision tree from scratch.

As stated before, using an oblivious decision tree may be attractive in this case as it adds features to a classifier in an incremental manner. Oblivious decision trees can be considered as restricted decision trees. For that reason any generalization error bound that has been developed for decision trees in the literature[31,32] can be used in this case also.

The following theorem introduces an upper bound and a lower bound of the VC dimension that was recently proposed by Dyulicheva.[33] For the sake of simplicity, the bound described in this section is developed assuming that the input attributes and the target feature are both binary. This bound can be extended for other cases in a straightforward manner. Note that each oblivious decision tree with nonbinary input attributes can be converted to a corresponding binary oblivious decision tree by using appropriate artificial features.

THEOREM 1. (Upper and lower bound for VC dimension of decision forest combined).[33] *The VC-dimension of ω decision trees with $\vec{T} = (t_1, \ldots, t_\omega)$ terminal nodes is*

$$\textit{not greater than} : rF\omega \log n - F\omega \log \frac{F\omega}{2}$$

and at least:$\max(\omega F, \log n)$*, where $F = \max_{i=1,\ldots,\omega}(t_i)$ and n is the number of input attribute (assuming binary attributes), r is the maximum permissible rank of forest's trees branches.*

3.4. Caching Mechanism

The Achilles heel of using genetic algorithm in feature set partitioning problem is that it requires the creation of a classifier for each subset in each solution candidate. Assuming that there are G generations, the population size is L, and that each solution has on average D subsets, then $G \cdot L \cdot D$ classifiers are created. Recall that by using oblivious decision trees we might not need to create from scratch each classifier, but reuse classifiers that have already created. It is well-known fact that one can trade computational complexity with storage complexity. Thus, we suggest using the following caching mechanism.

First of all when moving from one generation to the consequent generation, we can use all subsets that have remained without any change. By using the GWC operator and ignoring the mutation, each member in the new population has at least one subset (the anchor subset) that has not been changed at all. Moreover, all other subsets have some common members. In that case, we cannot use the oblivious decision tree as-is because the original tree might use attributes that are not used in the inherited subset. For this purpose, we eliminate attributes from the oblivious tree, layer-by-layer until we obtain an oblivious decision tree that all its attributes are used in the inherited subset.

Example. Assuming that two partitioning structures $Z^1 = \{\{a_2, a_4\}; \{a_5, a_3\}\}$ and $Z^2 = \{\{a_2, a_6\}; \{a_1, a_4, a_3\}\{a_5\}\}$ are given over the feature set $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$. We also assume that in the previous generation, the following attribute order has been used in the created oblivious decision trees: $a_4 \rightarrow a_2; a_5 \rightarrow a_3; a_2 \rightarrow a_6; a_1 \rightarrow a_3 \rightarrow a_4; a_5$.

Recall that by using the GWC operator (and ignoring the mutation operator), the following subsets may be obtained: $Z^3 = \{\{a_2, a_4\}; \{a_1, a_3\}; \{a_5\}; \{a_6\}\}$ and

$Z^4 = \{\{a_1, a_4, a_3\}; \{a_5\}; \{a_2\}\}$. Thus, to create the oblivious decision trees for Z^3 and Z^4, we can use the following oblivious decision trees as-is: $a_4 \rightarrow a_2; a_1 \rightarrow a_3 \rightarrow a_4$. The oblivious tree for $\{a_6\}$ will be created from scratch. The remaining subsets can be (partially or completely) obtained by removing attributes from the existing oblivious decision trees. The tree for $\{a_1, a_3\}$ can be obtained by removing attribute a_4 from $a_1 \rightarrow a_3 \rightarrow a_4$. This removal is possible as a_4 is located last. The tree for $\{a_2\}$ can be obtained by removing attribute a_6 from $a_2 \rightarrow a_6$.

In addition to using the oblivious decision trees of the previous generations, we can use the existing resemble in different subset of the current generation. While generating a decision tree, we check at the end of each iteration (i.e., after adding a new attribute to the decision tree) if there is another solution in the current generation that also group these features together in the same subset. If this is the case, we store the current decision tree in the cache for future use. Later on when time has come to generate the decision tree for the solution with the common subset, instead of creating the tree from scratch we use the tree that was stored in the caching mechanism. For example, if in the first generation we have the following members:

$$Z_1 = \{a_1, a_4, a_5, a_6\}; \{a_2; a_3; a_8; a_{10}\}; \{a_7; a_9\}$$

$$Z_2 = \{a_1, a_5, a_6, a_8\}; \{a_2; a_3; a_4; a_{10}\}; \{a_7; a_9\}$$

$$Z_3 = \{a_1, a_3, a_4, a_5, a_6\}; \{a_2; a_8; a_{10}\}; \{a_7; a_9\}$$

Assuming that we are evaluating the members one by one according to the above order, and that while creating the tree for the first subset in the first solution we got a decision tree with the following order $a_5 \rightarrow a_1 \rightarrow a_6$, then we might want to store this tree in the caching mechanism, and use it also for members 2 and 3.

It should be noted that using this caching mechanism reduce the search space, because it dictates the order in which the features are located in the decision tree. For instance in the last example the first tree of solution 2 could have the following structure: $a_8 \rightarrow a_1 \rightarrow a_5 \rightarrow a_6$. However, by using the tree $a_5 \rightarrow a_1 \rightarrow a_6$ that was stored in the cache, we ignore this structure in advance. To solve this dilemma, we decided no to store small trees (in this paper less than 3 attributes). In these cases, the saving in computational complexity is not worth the loss in generalization capability.

Obviously we would like to store in the cache the longest common subset. Thus, in each iteration we check if the current tree still can be used by the same number of solutions. If this is the case, the current tree will replace the older one.

3.5. Classification of an Unlabeled Instance

Following the creation of multiple oblivious decision trees, one can use them to classify an unlabeled instance, by performing the following steps:

(A) For each tree:
  1. Locate the appropriate leaf for the unseen instance.
  2. Extract the frequency vector (how many instances relate to each possible value of the target feature.)
  3. Transform the frequency vector to a probability vector according to Laplace's law of succession, as described in Section 2.
(B) Combine the probability vectors using the Naive Bayes combination.
(C) Select the target value maximizing the Naive Bayes combination.

4. EXPERIMENTAL STUDY

To illustrate the potential of feature set partitioning approach in classification problems and to evaluate the performance of the proposed genetic algorithm, a comparative experiment has been conducted on benchmark datasets. The following subsections describe the experimental set-up and the obtained results.

4.1. Algorithms Used

This study examines an implementation of genetic algorithm to feature set partitioning using the suggested adjacency matrix encoding, GWC operator and fitness function that is based on the VC dimension of multiple oblivious decision trees combined with Naive Bayes. This algorithm is called GDF (genetic algorithm-based decision forest). It uses population size of 100 chromosomes and it has been last for 500 generations.

The GDF algorithm is compared to our previous hill-climbing algorithm for feature set partitioning, denoted as DOG. The GDF algorithm is compared to the following single-model algorithms: IFN (A greedy oblivious decision tree inducer that uses gain ratio as the splitting criteria), Naive Bayes and C4.5. The first two unary-model algorithms were chosen as they represent specific points in the search space of the GDF algorithm. The C4.5 algorithm was chosen because it is considered as the state-of-the-art decision tree algorithm, which is widely used in many other comparative studies.

In the second part of the experiment, the proposed algorithm is also compared to AdaBoost, which is considered to be a general-purpose ensemble method. Finally, we compare our algorithm to the EROS algorithm that is also reduct-based ensemble. However, EROS is a nonmutually exclusive ensemble algorithm, i.e., it may use the same attribute in several classifiers of the ensemble.

4.2. Datasets Used

The selected algorithms were examined on 25 datasets, 22 of which have been selected manually from the UCI Machine Learning Repository.[34] The remaining datasets have been chosen from the NIPS2003 attribute selection challenge. The datasets chosen vary across a number of dimensions such as: the number of target classes, the number of instances, the number of input attributes, and their type (nominal, numeric). The information about the data is shown in Table III.

Table III. Data description.

Dataset	Number of instances	Number of features	Number of classes	Attribute characteristics
Arcene	100	10,000	2	Real
Audiology	200	70	9	Categorical
Aust credit	690	15	2	Categorical, nteger, eal
Bcan	286	10	2	Categorical
Dermatology	366	33	6	Categorical, Integer
Dexter	300	20,000	2	Real
Hepatitis	155	20	2	Categorical, Integer, Real
Ionosphere	351	34	2	Integer, Real
Iris	150	5	3	Real
Kr-vs-kp	3196	37	2	Categorical
Labor	57	17	2	Categorical
LED17	220	25	9	Categorical
Letter	15,000	17	26	Integer
Lung cancer	31	56	3	Integer
Madelon	2000	500	2	Integer
Mushroom	8124	22	2	Categorical
Nurse	12,960	8	5	Categorical
OPTIC	5628	64	10	Integer
Sonar	208	60	2	Real
Soybean	683	35	19	Categorical
Splice	1000	60	3	Categorical
Vote	290	16	2	Categorical
WDBC	569	32	2	Real
Wine	178	13	3	Integer, Real
Zoo	101	8	7	Categorical, Integer

4.3. Metrics Measured

In this experiment the following metrics were measured:

A. *Generalized Accuracy*: Represents the probability that an instance was classified correctly. To estimate the generalized accuracy, a 10-fold cross-validation procedure has been repeated five times. For each 10-fold cross validation, the training set was randomly partitioned into 10 disjoint instances subsets. Each subset was used once in a test set and nine times in a training set. The same cross-validation folds have been used for all algorithms. Since the average accuracy on the validation instances is a random variable, the confidence interval was estimated by using the normal approximation of the binomial distribution. Furthermore, the one-tailed paired t-test with a confidence level of 95% was used to verify whether the differences in accuracy between the GDF algorithm and other algorithms were statistically significant.
B. *Classifier Complexity*: As this paper focuses on decision trees, the classifier complexity was measured as the total number of nodes, including the leaves. For multiple decision trees classifiers, the complexity was measured as the total number of nodes in all trees.

The following additional metrics were measured to characterize the partitioning structures obtained by GDF algorithm:

A. Number of subsets.
B. Average number of features in a single subset.

4.4. Comparing to Single-Model Algorithms

Table IV presents the results obtained by averaging five standard 10-fold cross-validation experiments. The results indicates that there is no significant case where Naive Bayes or IFN were more accurate than GDF, on the other hand GDF was significantly more accurate than Naive Bayes and IFN in 16 databases and 14 databases, respectively. Moreover, GDF was significantly more accurate than C4.5 in 13 databases, and less accurate in only two databases.

The results of the experimental study are encouraging. On the datasets obtained from the UCI repository, the GDF outperformed Naive Bayes mostly when the data were large in size or had a small number of features. For moderate dimensionality (from 50 features up to 500), the performance of Naive Bayes was not necessarily inferior. More specifically, regarding the datasets: OPTIC, SONAR, SPI, AUDIOLOGY, LUNG-CANCER, only in three of the datasets (SPI, AUDIOLOGY, and LUNG-CANCER), was the superiority of GDF over Naive Bayes statistically significant. However, for high-dimensionality datasets (having at least 500 features), GDF significantly outperforms Naïve Bayes in all cases.

Analyzing the number of features in each subset shows that the GDF algorithm tends to build small subsets. Moreover, there is one case (OPTIC) in which the GDF algorithm used only one feature in each tree. In these cases the classifiers built are equivalent to Naive Bayes, besides the fact that not all input attributes are necessarily used (nonrelevant features might be dropped). This suggests that in some cases GDF acts as a attribute selection procedure for Naive Bayes.

4.5. Comparing to other Ensemble Algorithms

In this section, we compare GDF to AdaBoost and EROS. Since the accuracy and the model complexity are affected by the ensemble size (number of classifiers), we have examined various ensemble sizes (from 1 to 25). Table V presents the obtained results.

The results indicate that there are datasets in which GDF algorithm obtained accuracy similar to the accuracies obtained by AdaBoost (like in the case of AUST dataset). There are cases in which AdaBoost has achieved much higher accuracies (like in the cases of AUDIOLOGY and HEPATITIS). There are cases in which GDF achieved the best accuracies (like in the case of BCAN or MADELON).

Analyzing the results of the entire datasets collection indicates that in seven datasets AdaBoost achieved significantly higher accuracies (note that the compared value is the best accuracy achieved by enumerating the ensemble size from 1 to 25). On the other hand, GDF was significantly more accurate than AdaBoost in only four datasets including the high-dimensional datasets: MADELON and DEXTER.

The above results disregard the model complexity. For instance, AdaBoost obtained an accuracy of 87.72% for the LETTER dataset (comparing to accuracy of 75.02% obtained by GDF algorithm). Nevertheless, the average complexity of AdaBoost model in this case was 240319 nodes (comparing to only 313 nodes of GDF in this case).

Table IV. Summary of experimental results.

Dataset	Naive Bayes	C4.5		IFN		GDF			
	Accuracy	Accuracy	Number of nodes	Accuracy	Number of nodes	Accuracy	Number of nodes	Number of subsets	Average subset size
Arcene	$^{+}70 \pm 12.3$	75 ± 9.2	9	$^{+}54 \pm 8.3$	46	76 ± 6.2	116	8	7
Audiology	$^{+}65.5 \pm 7.39$	$^{+}75 \pm 6.95$	52	$^{+}74 \pm 7.95$	100	82.2 ± 4.29	129	7	2.2
Aust credit	84.93 ± 2.7	85.36 ± 5.1	30	84.49 ± 5.1	27	84.45 ± 4.6	59	3	3.2
Bcan	97.4249 ± 1.17	$^{+}92.99 \pm 2.87$	61	$^{+}94.39 \pm 3.5$	55	96.43 ± 1.6	75	5	1.1
Dermatology	$^{+}72.06 \pm 2.1$	$^{+}74.27 \pm 3.3$	17	$^{+}72.44 \pm 3.9$	26	92.15 ± 2.2	74	3	4
Dexter	$^{+}86.33 \pm 3.9$	$^{+}78.33 \pm 3.6$	53	$^{+}76.13 \pm 2.1$	47	91.18 ± 2.2	783	16	52.5
Hepatitis	82.58 ± 7.56	81.29 ± 5.46	7	78.97 ± 8.99	68	82.29 ± 5.6	7	1	2.9
Ionosphere	$^{+}86.31 \pm 0.2$	$^{+}85.92 \pm 1.3$	22	$^{+}82.32 \pm 2.4$	27	91.42 ± 1.9	67	5	3.2
Iris	95.33 ± 5.05	96 ± 3.33	11	96 ± 3.33	90	96 ± 3.33	11	1	3.8
Kr-vs-kp	$^{+}87.86 \pm 1.41$	99.44 ± 0.55	87	98.06 ± 0.42	220	99.32 ± 0.75	134	3	7.7
Labor	92.98 ± 4.52	$^{+}73.72 \pm 12.72$	12	$^{+}84.63 \pm 8.14$	32	94.21 ± 3.5	21	4	2
LED17	$^{+}63.18 \pm 8.7$	$^{+}59.09 \pm 6.9$	69	$^{+}55.55 \pm 6.3$	73	71.62 ± 2.6	46	4	3.3
Letter	73.29 ± 1	74.96 ± 0.8	11169	$^{+}69.56 \pm 0.7$	5321	74.25 ± 1.7	311	10	1.6
Lung cancer	$^{+}41.94 \pm 19.96$	$^{+}38.71 \pm 17.82$	16	$^{+}38.71 \pm 17.82$	16	55.15 ± 10.05	26	5	2
Madelon	$^{+}58.3 \pm 1.5$	69.8 ± 4.7	259	$^{+}62 \pm 3.4$	127	72.1 ± 2.9	997	3	101.7
Mushroom	$^{+}95.48 \pm 0.9$	100 ± 0	28	100 ± 0	30	100 ± 0	38	1	4.9
Nurse	$^{+}65.39 \pm 24$	97.45 ± 0.4	527	92.47 ± 0.5	135	95.01 ± 1.16	342	2	3.9
OPTIC	91.73 ± 1.3	$^{+}62.42 \pm 2$	4059	$^{+}48.90 \pm 2.5$	1257	92.84 ± 1.1	968	57	1
Sonar	75.48 ± 7.3	$^{+}69.71 \pm 5.4$	51	76.48 ± 6.8	97	76.71 ± 2.9	125	5	2.3
Soybean	$+91.95 \pm 1.99$	$^{+}92.83 \pm 1.52$	85	92.24 ± 2.46	72	93.15 ± 1.2	133	2	4.8
Splice	$^{+}94.1 \pm 0.4$	$^{+}91.2 \pm 1.9$	117	$^{+}87.00 \pm 2.6$	523	95.3 ± 1.3	406	15	2.9
Vote	$^{+}90.34 \pm 3.44$	$^{-}96.21 \pm 2.45$	16	93.79 ± 2.8	23	92.62 ± 2.2	24	1	6.8
WDBC	$^{+}86.72 \pm 3.2$	$^{+}88.13 \pm 4.2$	71	$^{+}89.13 \pm 2.5$	67	95.4 ± 3.1	189	5	5.3
Wine	96.63 ± 3.9	$^{+}85.96 \pm 6.9$	41	$^{+}91.45 \pm 5$	41	$94. \pm 3.41$	65	5	1.8
Zoo	$^{+}89.11 \pm 7$	$^{+}93.07 \pm 5.8$	21	$^{+}90.89 \pm 9.7$	21	96.21 ± 2.42	19	3	2.4

The superscript "+" indicates that the accuracy rate of GDF was significantly higher than the corresponding algorithm at confidence level of 95%. The "–" superscript indicates the accuracy was significantly lower.

Table V. Summary of experimental results.

	AdaBoost			EROS			DOG				GDF			
Dataset	Accuracy	Number of nodes	Ensemble size	Accuracy	Number of nodes	Ensemble size	Accuracy	Number of nodes	Number of subsets	Average subset size	Accuracy	Number of nodes	Number of subsets	Average subset size
Arcene	78 ± 5.2	467	10	$^{-}79 \pm 4.3$	471	10	76 ± 8.1	97	12	3.2	76 ± 6.2	116	8	7
Audiology	83.5 ± 4.25	471.2	8	82.5 ± 4.88	452	8	$^{+}78.5 \pm 6.54$	64	3	4.67	82.2 ± 4.29	129	7	2.2
Aust credit	85.36 ± 3.6	30	1	86.36 ± 2.65	42	3	86.52 ± 2.5	84	11	1.27	84.45 ± 4.6	59	3	3.2
Bcan	96.71 ± 2.7	1793	19	96.71 ± 2.6	1878	19	97.42 ± 1.17	99	9	1	96.43 ± 1.6	75	5	1.1
Dermatology	97.17 ± 2.1	98	12	98.36 ± 2.9	128	15	93.06 ± 2.7	92	5	3.2	92.15 ± 2.2	74	3	4
Dexter	$^{+}81.13 \pm 3.1$	391	7	$^{+}82.13 \pm 3.1$	380	7	89.33 ± 2.7	562	11	52.72	91.18 ± 2.2	783	16	52.5
Hepatitis	81.29 ± 5.46	7	1	83.29 ± 5.46	27	4	80 ± 6.89	38	2	2	82.29 ± 5.6	7	1	2.9
Ionosphere	94.72 ± 3.2	115	9	95.73 ± 2.8	172	12	91.42 ± 1.9	67	5	3.2	91.42 ± 1.9	67	5	3.2
Iris	96 ± 3.33	11	1	95 ± 3.33	11	4	95.33 ± 5.05	40	4	1	96 ± 3.33	11	1	3.8
Kr-vs-kp	99.69 ± 0.59	421	5	99.45 ± 1.05	418	7	98.47 ± 0.63	330	2	7	99.32 ± 0.75	134	3	7.7
Labor	$^{-}100 \pm 0$	59	5	$^{-}100 \pm 0$	58	4	96.49 ± 5.5	67	16	1	94.21 ± 3.5	21	4	2
LED17	65.91 ± 4.2	365.8	5	71.91 ± 3.9	351	12	73.64 ± 5.5	370	7	3.28	71.62 ± 2.6	46	4	3.3
Letter	$^{-}87.72 \pm 2.3$	240319	20	$^{-}89.72 \pm 2.3$	151874	16	73.46 ± 0.64	272	16	1	74.25 ± 1.7	311	10	1.6
Lung cancer	$^{-}57.5 \pm 12.82$	32.8	3	$^{-}57.5 \pm 12.7$	62	4	53.55 ± 10.05	27	4	2	55.15 ± 10.05	26	5	2
Madelon	$^{+}67.77 \pm 4.14$	3693	14	$^{+}69.77 \pm 3.9$	3537	14	71.4 ± 2.6	660	2	117.8	72.1 ± 2.9	997	3	101.7
Mushroom	100 ± 0	30	1	100 ± 0	824	12	100 ± 0	28	1.2	7.67	100 ± 0	38	1	4.9
Nurse	$^{-}98.2 \pm 1.5$	6069	19	$^{-}97.9 \pm 1.5$	6013	19	$^{+}91.65 \pm 0.6$	38	6	1.33	95.01 ± 1.16	342	2	3.9
OPTIC	$^{+}87.24 \pm 2.1$	73838	20	$^{+}87.24 \pm 1.6$	71221	19	91.73 ± 1.4	981	64	1	92.84 ± 1.1	968	57	1
Sonar	$^{-}79.24 \pm 6.7$	994	16	84.62 ± 7.3	1170	12	77.12 ± 8.7	98	35	1.657	76.71 ± 2.9	125	5	2.3
Soybean	93.47 ± 2.51	1271	15	93.98 ± 1.9	1294	15	92.9 ± 2.56	122	3	4	93.15 ± 1.2	133	2	4.8
Splice	$^{+}93.7 \pm 4.6$	2331	19	94.9 ± 4.9	2293	19	95.8 ± 0.9	300	50	1.2	95.3 ± 1.3	406	15	2.9
Vote	$^{-}96.21 \pm 2.3$	16	1	$^{-}96.65 \pm 2.5$	673	12	$^{+}90.52 \pm 1.23$	18	6	1.333	92.62 ± 2.2	24	1	6.8
WDBC	96.12 ± 3	420	5	$^{-}98.24 \pm 2.1$	541	7	95.6 ± 3.1	196	6	4.7	95.4 ± 3.1	189	5	5.3
Wine	95.56 ± 6.1	513	11	97.21 ± 4.2	685	20	96.63 ± 3.9	143	13	1	$94. \pm 3.41$	65	5	1.8
Zoo	$^{-}100 \pm 0$	110	7	$^{-}100 \pm 0$	113	7	98.02 ± 3.02	50	4	4	96.21 ± 2.42	19	3	2.4

The superscript "+" indicates that the accuracy rate of GDF was significantly higher than the corresponding algorithm at confidence level of 95%. The "–" superscript indicates the accuracy was significantly lower.

Taking into consideration the model's complexity, we have compared the accuracy obtained by the AdaBoost algorithm to that of GDF algorithm using the same complexity of the GDF's model. Because it is impossible to tune the AdaBoost model's complexity to a certain value, we use interpolation of the two closest points in the AdaBoost's accuracy–complexity graph that bounds this value, on condition that these points are "dominant," namely, there are no less complicated points in the AdaBoost's graph that has higher accuracy. Geometrically this examine on what datasets the GDF point is significantly above or below the AdaBoost's trend line. If no such pair of points could be found we used the highest accuracy that its complexity is less or equal to the GDF's model complexity. If no such point can be found, we used the first point (ensemble of size one).

The accuracy–complexity tradeoff analysis indicates that GDF has significantly outperformed AdaBoost in 13 datasets while AdaBoost has significantly outperformed DOG in only two (VOTE, LABOR) datasets, two of which the complexity of AdaBoost was much higher than the DOG complexity (because the single C4.5 decision tree has already contained more nodes than the GDF classifiers), namely AdaBoost is not necessary better in these cases because GDF introduces new points in the complexity–accuracy tradeoff. Furthermore, in the VOTE dataset, a single C4.5 has already significantly outperformed the GDF algorithm. This observation might imply that the limited structure of oblivious decision trees used in GDF algorithm as oppose to the C4.5 decision tree used in AdaBoost to might be the origin to the poor results in these cases.

Furthermore GDF has obtained better accuracy–complexity tradeoff than AdaBoost for all datasets with moderate dimensionality (number of features between 50 and 100) and with high dimensionality (number of features greater than 100).

The GDF has another important advantage over AdaBoost. The decision trees in GDF are based on the original distribution of the training set. The class distribution at the tree's leafs is supported by the training set. In Adaboost (starting from the second decision tree), the class distribution at the leaf level does not necessarily fit the original distribution, what makes it difficult justifying the results to a nonprofessional user.

Analyzing the results of the entire datasets collection indicates that in eight datasets EROS achieved significantly higher accuracies (note that the compared value is the best accuracy achieved by enumerating the ensemble size from 1 to 25). On the other hand, GDF was significantly more accurate than EROS in only three datasets. The GDF has two important advantages on EROS:

1. *Simplicity*: It usually creates a smaller decision trees. Smaller decision trees are considered to be more comprehensive to users.
2. *Comprehensibility*: The mutually exclusive property enhances the comprehensibility and the ease of usage. Consider a case in which data mining is used for improving the quality of a certain manufacturing line. In this case, the target attribute stands for the quality of a certain product (high/low) and the input attributes represent the values of various manufacturing parameters (such as speed, temperature, etc.). In mutually exclusive decision trees, the user can easily find the best parameters values by selecting in each decision tree the path that most favors the "high" label (i.e., with the highest probability). If the

mutually exclusive property is not kept (like in EROS) finding the best parameters values become more complicated since paths from different trees might incorporate the same attributes but not necessarily with the same values.

Comparing the accuracy of GDF and DOG indicated that in most of the cases GDF has obtained better results. This observation is not surprising, considering the fact that GDF performs more intensive search than DOG. Nevertheless, DOG's model complexity (total number of nodes) was comparable to the complexity obtained by C4.5 algorithm in most of the cases. Comparing the mean number of subsets obtained by DOG (11.88) and that obtained by GDF (6.96) indicates that DOG tends to have more subsets. Moreover, in 17 datasets out of 25 datasets DOG has incorporated more features than GDF. However, for high-dimensionality datasets (having at least 500 features), GDF has significantly used more features than DOG.

4.6. The Effectiveness of Using VC-Dimension Bounds for GA Fitness Function

The aim of this section is to examine the performance of using the above mentioned VC-dimensions bounds as oppose to the wrapper approach that is usually

Table VI. Summary of experimental results.

Dataset	GA with wrapper approach	GDF with VC lower bound	GDF with VC upper bound
Arcene	73.43 ± 6.60	$^{+}$71.87 ± 6.73	76 ± 6.2
Audiology	81.68 ± 3.89	$^{+}$77.25 ± 4.09	82.2 ± 4.29
Aust credit	86.52 ± 2.60	$^{+}$83.83 ± 4.20	84.45 ± 4.6
Bcan	96.82 ± 1.18	96.67 ± 1.00	96.43 ± 1.6
Dermatology	$^{-}$96.27 ± 3.2	93.01 ± 2.7	92.15 ± 2.2
Dexter	91.09 ± 1.51	90.23 ± 2.20	91.18 ± 2.2
Hepatitis	83.67 ± 5.41	$^{+}$76.15 ± 5.09	82.29 ± 5.6
Ionosphere	94.86 ± 4.2	91.66 ± 1.6	91.42 ± 1.9
Iris	94.87 ± 3.34	94.47 ± 3.38	96 ± 3.33
Kr-vs-kp	99.35 ± 0.31	99.27 ± 0.63	99.32 ± 0.75
Labor	95.64 ± 3.66	94.54 ± 3.66	94.21 ± 3.5
LED17	73.67 ± 3.20	70.97 ± 3.69	71.62 ± 2.6
Letter	$^{-}$77.34 ± 1.11	74.72 ± 1.01	74.25 ± 1.7
Lung cancer	50.03 ± 10.02	$^{+}$45.74 ± 10.32	55.15 ± 10.05
Madelon	71.87 ± 2.38	$^{+}$67.59 ± 3.14	72.1 ± 2.9
Mushroom	100.00 ± 0	100.00 ± 0	100 ± 0
Nurse	96.85 ± 1.15	$^{+}$93.35 ± 0.95	95.01 ± 1.16
OPTIC	91.91 ± 0.88	91.11 ± 0.79	92.84 ± 1.1
Sonar	75.60 ± 2.68	75.20 ± 2.71	76.71 ± 2.9
Soybean	95.04 ± 0.53	94.57 ± 0.27	93.15 ± 1.2
Splice	96.31 ± 0.16	96.00 ± 0.53	95.3 ± 1.3
Vote	94.16 ± 2.56	$^{+}$90.99 ± 2.92	92.62 ± 2.2
WDBC	96.32 ± 2.8	94.9 ± 2.8	95.4 ± 3.1
Wine	94.27 ± 3.90	93.98 ± 4.25	94. ± 3.41
Zoo	98.69 ± 2.72	97.02 ± 2.79	96.21 ± 2.42

The superscript "+" indicates that the accuracy rate of GDF with VC upper bound was significantly higher than the corresponding algorithm at confidence level of 95%. The "–" superscript indicates the accuracy was significantly lower.

used for calculating the fitness function in GAs. The wrapper approach usually provides a better approximation to the generalization error then theoretical methods. However, it adds considerable overhead to an already expensive search process. Table VI presents the results obtained by using the VC upper bound, VC lower bound, and the wrapper approach. The results indicate that the wrapper and the upper VC dimension bound have quite similar results. In fact only in two cases (letter and dermatology) does the superiority of the wrapper approach was statistically significant. Thus, it is possible to conclude that for the use of GAs fitness function one can use a less accurate approximation than the wrapper approach and still gets similar results. However, examining the results obtained by the VC lower bound indicate that there are eight cases in which the upper bound was significantly better. Thus, it is empirically evident that the lower bound is too rough to be used as a basis for the GAs' fitness function.

5. CONCLUSIONS

This paper presents a new genetic algorithm for constructing a decision forest which combining disjoints classification trees. The basic idea is to decompose the original set of features into several subsets, build a decision tree for each projection, and then combine them.

This paper examine if genetic algorithm can be useful for discovering the appropriate partitioning structure. For this purpose, we have suggested a new encoding schema and fitness function that was specially designed for feature set partitioning with oblivious decision trees. Additionally, a caching mechanism has been implemented to reduce computational complexity.

The proposed framework was tested on 25 datasets. The results show that this algorithm outperforms our previous hill-climbing feature set partitioning algorithm. Furthermore, this algorithm tends to outperform other ensemble methods in accuracy–complexity trade-off. This observation leads to the conclusion that the proposed algorithm can be used for creating compact ensemble structures.

Additional issues to be further studied include: examining how the feature set partitioning concept can be implemented using other inducers like neural networks and by examining other techniques to combine the generated classifiers (like voting).

References

1. Rokach L, Maimon O. Top down induction of decision trees classifiers: A survey. IEEE SMC Trans C 2005;35(4):476–487.
2. Murphy PM, Pazzani MJ. Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction. J Artif Intell Res 1994;1(1):257–275.
3. Tong W, Hong H, Fang H, Xie Q, Perkins R. Decision forest: Combining the predictions of multiple independent decision tree models. J Chem Inf Comput Sci 2003;43(2):525–531.
4. Hong H, Tong W, Perkins R, Fang H, Xie Q, Shi L. Multiclass decision forest—a novel pattern recognition method for multiclass classification in microarray data analysis. DNA Cell Biol 2004;23(10):685–694.

5. Xie Q, Ratnasinghe L, Hong H, Perkins R, Tang Z, Hu N, Taylor P, Tong W. Decision forest analysis of 61 single nucleotide polymorphisms in a case–control study of esophageal cancer; a novel method. BMC Bioinform 2005;6(Suppl 2):S4.
6. Jimenez LO, Landgrebe DA. Supervised classification in high-dimensional space: Geometrical, statistical, and asymptotical properties of multivariate data. IEEE Trans Syst Man, Cybernet C Appl Rev 1998;28:39–54.
7. Fukunaga K. Introduction to statistical pattern recognition. San Diego, CA: Academic; 1990.
8. Hwang, J, Lay S, Lippman A. Nonparametric multivariate density estimation: A comparative study. IEEE Trans Signal Process 1994;42(10):2795–2810.
9. Bellman R. Adaptive control processes: A guided tour. Princeton, NJ: Princeton University Press; 1961.
10. Liu H, Motoda H. Attribute selection for knowledge discovery and data mining. Boston, MA: Kluwer; 1998.
11. Kusiak A. Decomposition in data mining: An industrial case study. IEEE Trans Electr Packag Manuf 2000;23(4):345–353.
12. Maimon O, Rokach L. (2005), Decomposition methodology for knowledge discovery and data mining: Theory and applications, series in machine perception and artificial intelligence, Vol. 61. Singapore: World Scientific Publishing, 2005.
13. Rokach L, Maimon O. Feature set decomposition for decision trees. J Intell Data Anal 2005;9(2):131–158.
14. Freitas A. Evolutionary algorithms for data mining. In: O. Maimon, L. Rokach editors. The data mining and knowledge discovery handbook. Berlin: Springer; 2005. pp 435–467.
15. Opitz D. Attribute selection for ensembles. In: Proc 16th National Conf on Artificial Intelligence. Menlo Park, CA: AAAI, 1999. pp 379–384.
16. Sharpe PK, Glover RP. Efficient GA based techniques for classification. Appl Intell 1999;11:277–284.
17. Kudo M, Sklansky J. Comparison of algorithms that select features for pattern classifiers. Pattern Recogn 2000;33:25–41.
18. Hsu WH. Genetic wrappers for attribute selection in decision tree induction and variable ordering in Bayesian network structure learning. Inform Sci 2004;163(1–3):103–122.
19. Freund Y, Schapire R. Experiments with a new boosting algorithm. In: Machine Learning: Proc for the 13th Intl Conf. San Francisco, CA: Morgan Kaufmann; 1996. pp 148–156.
20. Hu Q, Yu D, Xie Z, Li X. EROS: Ensemble rough subspaces. Pattern Recogn 2007;40(12):3728–3739.
21. Chawla NV, Hall LO, Bowyer KW, Kegelmeyer WP. Learning ensembles from bites: A scalable and accurate approach. J Machine Learn Res 2004;5:421–451.
22. Opitz D, Shavlik J. Actively searching for an effective neural-network ensemble. Connect Sci 1996;8(3/4):337–353.
23. Hsu WH, Welge M, Wu J, Yang T. Genetic algorithms for selection and partitioning of attributes in large-scale data mining problems. In: Proc of the Joint AAAI-GECCO Workshop on Data Mining with Evolutionary Algorithms, Orlando, FL, July 1999.
24. Rokach L. Genetic algorithm-based feature set partitioning for classification problems, Pattern Recog 2008;41(5):1676–1700.
25. Domingos P, Pazzani M. On the optimality of the Naive Bayes classifier under zero-one loss. Machine Learn 1997;29(2):103–130.
26. Duda R, Hart P. Pattern classification and scene analysis. New-York: Wile; 1973.
27. Louis SJ, Rawlins GJE. Predicting convergence time for genetic algorithms. In: LD, Whitley editor. Foundations of genetic algorithms 2. San Francisco, CA: Morgan Kaufmann; 1993. pp 141–161.
28. Almuallim H, Dietterich TG. Learning Boolean concepts in the presence of many irrelevant features. Artif Intell 1994;69(1–2):279–306.
29. Schlimmer JC. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In: Proc 1993 Int Conf Machine Learning. San Mateo, CA: Morgan Kaufmann; 1993. pp 284–290.

30. Langley P, Sage S. Induction of selective Bayesian classifiers. In: Proc 10th Conf on Uncertainty in Artificial Intelligence. Seattle, WA: Morgan Kaufmann; 1994. pp 399–406.
31. Simon H. The Vapnik-Chervonenkis dimension of decision trees with bounded rank. Inform Process Lett 1991;39(3):137–141.
32. Mansour Y, McAllester D. Generalization bounds for decision trees. In: Proc 13th annual conf Computer Learning Theory, San Francisco, CA: Morgan Kaufmann; 2000. pp 69–80.
33. Dyulicheva Y. Bound for the VSD of an r-reduced empirical forest. Tavrich Vest Inform Mat 2003;1:31–42.
34. Merz CJ, Murphy PM. UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science; 1998.
35. Estabrooks A, Jo T, Japkowicz N. A Multiple resampling method for learning from imbalances data sets. Comput Intell 2004;20(1):18–36.
36. Kearns M, Li M, Valiant L. Learning boolean formulas. J ACM 1994;41(6):1298–1328.
37. Opitz D, Maclin R. Popular ensemble methods: An empirical study. J Artif Res 1999;11:169–198.
38. Quinlan JR. C4.5: Programs for machine learning. San Francisco, CA: Morgan Kaufmann; 1993. 40. Rokach L, Maimon O. Data mining for improving the quality of manufacturing: A feature set decomposition approach. J Intell Manuf 2006;17(3):285–299.
39. Rand WM. Objective criteria for the evaluation of clustering methods. J Am Statist Assoc 1971;66:846–850.