Knowledge-Based Systems 110 (2016) 86–97

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm

Md Nasim Adnan*, Md Zahidul Islam

School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia

ARTICLE INFO

Article history:
Received 15 March 2016
Revised 31 May 2016
Accepted 9 July 2016
Available online 9 July 2016

Keywords:
Classification
Decision tree
Decision forest
Random forest
Ensemble accuracy

ABSTRACT

A decision forest is an ensemble of decision trees, and it is often built to discover more patterns (i.e. logic rules) and predict/classify class values more accurately than a single decision tree. Existing decision forest algorithms are typically used for building huge numbers of decision trees, involving large memory and computational overhead, in order to achieve high accuracy. Generally, many of the trees do not contribute to improving the ensemble accuracy of a forest. As a result, ensemble pruning algorithms aim to get rid of those trees while generating a subforest in order to achieve higher (or comparable) ensemble accuracy than the original forest. The objectives are two fold: select as small number of trees as possible, and maintain the ensemble accuracy of the subforest as high as possible. An optimal subforest can be found by exhaustive search; however it is not practical for any standard-sized forest as the number of candidate subforests grows exponentially. In order to avoid the computational burden of an exhaustive search, many greedy and genetic algorithm-based subforest selection techniques have been proposed in literature. In this paper, we propose a subforest selection technique that achieves small size as well as high accuracy. We use a genetic algorithm where we carefully select high quality individual trees for the initial population of the genetic algorithm in order to improve the final output of the algorithm. Experiments are conducted on 20 data sets from the UCI Machine Learning Repository to compare the proposed technique with several existing state-of-the-art techniques. The results indicate that the proposed technique can select effective subforests which are significantly smaller than original forests while achieving better (or comparable) accuracy than the original forests.

1. Introduction

The use of ensembles in classification have been actively studied in recent years [2,15,28,39,50]. Interestingly, an ensemble of classifiers is found to be effective for unstable classifiers such as decision trees [43]. Decision trees are considered to be an unstable classifier because a slight change in a training data set can cause a significant change between the resulting decision trees obtained from the original and modified data sets [43]. A decision forest is an ensemble of decision trees where an individual decision tree acts as a base classifier. The classification is performed by taking a vote based on the predictions made by each decision tree of the decision forest [43].

In order to achieve better ensemble accuracy a decision forest needs both accurate and diverse (in terms of classification errors) individual decision trees as base classifiers [17,37]. An accurate decision tree can be generated by feeding a training data set to a decision tree algorithm such as CART [10]. Nevertheless, a single decision tree can discover only one set of logic rules and thus may wrongly predict the class value of a test record which could have been predicted correctly by a more appropriate logic rule. A different decision tree can be obtained from a differentiated data set which may include a more appropriate logic rule for the given test record. If a decision forest contains a set of decision trees which are different from each other then some of the trees may discover appropriate logic rules for a set of test records while some other trees may discover appropriate logic rules for another set of test records, resulting in better generalization performance for the forest.

Accordingly, it is easy to comprehend that classifiers to be combined should be as different/diverse as possible; nonetheless if they were identical, there could be little or no improvement by combining them [41]. For example, an ensemble of three identical classifiers with 95% individual accuracy is worse than the ensemble of three classifiers being least correlated in classification errors with 67% individual accuracy [51]. It is proved that, diverse classifiers can lead to uncorrelated errors [22]. However, to

* Corresponding author.
E-mail addresses: madnan@csu.edu.au (M.N. Adnan), zislam@csu.edu.au (M.Z. Islam).

http://dx.doi.org/10.1016/j.knosys.2016.07.016
0950-7051/

establish the scope of generating too diverse classifiers may be the cause of generating less accurate classifiers as optimization on the two conflicting objectives can not be attained simultaneously [17]. In literature, we find a considerable study on the accuracy-diversity trade off and find that accuracy does not need to be ignored for diversity [7,23,44]. Yet, till date no data set (problem) independent relationship exists between individual accuracy and diversity which is optimal for achieving high ensemble accuracy [41,48].

Several decision forest algorithms exist aiming to generate more accurate and diverse decision trees by manipulating the training data set. Bagging [8] generates a new training data set $\boldsymbol{D}_i$ where the records of $\boldsymbol{D}_i$ are chosen randomly from the original training data set $\boldsymbol{D}$. A new training data set $\boldsymbol{D}_i$ contains the same number of records as in $\boldsymbol{D}$. Thus, some records of $\boldsymbol{D}$ can be chosen multiple times and some records may not be chosen at all. This approach of generating a new training data set is known as bootstrap sampling. On an average 63.2% of the original records are typically chosen in a bootstrap sample [16]. Bagging generates a predefined number (T) of bootstrap samples $\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_T$ using the above approach. A decision tree building algorithm is then applied on each bootstrap sample $\boldsymbol{D}_i$ $(i = 1, 2, \ldots, T)$ in order to generate altogether T number of trees for the forest. The Random Subspace algorithm [17] randomly draws a subset of attributes (subspace) $\boldsymbol{f}$ from the entire attribute set $\boldsymbol{m}$ in order to determine the splitting attribute for each node of a decision tree. Random Forest [9] is regarded as a state-of-the-art decision forest building algorithm [5,6] that is technically a combination of Bagging [8] and Random Subspace [17] algorithms.

Despite being structurally different, every rule cannot be guaranteed to be least correlated in terms of classification error. As a result, a major drawback of a decision forest is that, it requires a large number of trees to ensure convergence of the ensemble effect [14,32]. Having a large number of trees entails to large memory and computational overhead [36]. For example, a 100-tree Random Forest generated from 51 KB "Car Evaluation" data set (one of the most popular data sets in the UCI Machine Learning Repository [26]) may require 795 KB of memory. Other than the memory requirement, a test record needs to be evaluated by the 795 KB Random Forest which incurs a considerable computational cost for just one prediction. These aspects can be crucial for online applications such as stock market and weather forecasting [36]. Besides, all trees of a forest are not equally contributing; some of them may amplify other's wrong predictions to downgrade the overall predictive performance of the forest. Thus, a subset of trees (subforest) obtained through pruning some relatively detrimental trees often perform better than the complete forest [29,30,32,33,36,53].

Assuming that we have a forest with T trees, there can be at most $2^T - 1$ non empty subforests. The most reliable strategy for finding the optimal subforest among $2^T - 1$ candidates is to evaluate them all on a training data set as it is shown that maximizing the ensemble accuracy on training data set may lead to improving the generalization performance [42]. However, computing the optimal subforest by exhaustive search is unrealisable for a typical 100-tree forest as the number of candidate subforests grows exponentially.

In order to avoid the computational burden of the exhaustive search, a lot of greedy approaches have been proposed in literature [29,30,33,34,36,48]. Some of these methods are heuristic in nature that tend to sort individual trees in an order based on their individual contribution. Subforests are then constructed by selecting a user defined number of trees according to the order [29,30,36,48]. Some other approaches employ general hill climbing strategy that starts with an initial forest (empty or full) and then adds/removes a single tree in order to increase forest accuracy or diversity [30,33,34]. The same kind of hill climbing strategy is also taken to increase the average of individual tree accuracy or diversity [30]. In general, greedy approaches are said to be computationally inexpensive yet they can get stuck in a local optima and generate sup-optimal subforest. To overcome this complication, it is possible to use genetic algorithms (GA) that with high probability can select optimal/near-optimal subforest. Despite being computationally intensive, these algorithms are no longer exponential to the size of the original forest [32].

There is a number of GA-based ensemble pruning techniques [20,40,53]. Unlike some greedy approaches, GA-based techniques require no user defined size for a subforest. Nevertheless, to be effective a subforest needs to include high quality individual trees and exclude low quality trees. Now, an interesting question is "*What is a High-Quality tree in the context of a forest*"? As described earlier, it is insufficient to use either individual accuracy or diversity to evaluate a tree for a forest. Any ensemble that is made of classifiers that are simultaneously accurate and diverse, is said to have better generalization performance [13,44]. Hence, subforest selection approaches should endeavour to include trees that are relatively more accurate and diverse. However, GA-based techniques proposed so far randomly select trees from the original forest as the initial population. As a result, the chosen initial population may not be of high quality. This may lead to difficulty in finding optimal/near-optimal subforest in a limited number of iterations. Good quality initial population typically lead to a good quality final solution in a GA [38].

In this paper, we present a novel method for selecting simultaneously accurate and diverse individual trees that make up the initial population of a GA which finally produces an effective subforest. To the best of our knowledge, no previous techniques so far have been proposed to include optimized trees in any form for initial population of a GA-based subforest selection technique. Feeding relatively more accurate and diverse trees as initial population in GA has proved to be very effective as indicated by the experimental results of this study on 20 data sets that are publicly available from the UCI Machine Learning Repository [26].

The rest of this paper is organized as follows: In Section 2 we discuss some of the well-known subforest selection/forest pruning algorithms related to the proposed method. Section 3 explains the motivation to solve the subforest problem using GA. The proposed GA-based subforest selection algorithm is described in Section 4. Section 5 discusses the experimental results in detail. Finally, we offer some concluding remarks in Section 6.

2. Related works

In a real-life scenario, it is not practical to generate a classifier of size 795 KB from 51 KB "Car Evaluation" data set when a simple classifier such as nearest neighbour can perform reasonably well. Motivated by a similar fact, Margineantu and Dietterich [30] proposed a number of heuristic-based forest pruning techniques for AdaBoost. One notable pruning technique is Kappa Pruning where they chose diverse trees according to Kappa value. Kappa typically estimates the diversity between *two trees* T_i and T_j. Diversity among *more than two trees* is computed by first computing the Kappa (K) value of a single tree T_i with the ensemble of trees except the tree in consideration (i.e. with $F - T_i$ where F is the set of all trees in the forest) [2]. The combined prediction of the ensemble (computed through the majority voting) can be regarded as a single tree T_j. Then Kappa is computed between T_i and T_j as shown in Eq. (1), where $Pr(a)$ is the probability of the observed agreement between two trees T_i and T_j, and $Pr(e)$ is the probability of the random agreement between T_i and T_j.

$$K = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{1}$$

According to Eq. (1), when $K = 1$ two trees agree on every example. When $K = 0$ they disagree on every examples except agreements by chance. Rarely, K can be negative when two trees disagree in most examples suppressing agreements by chance. Thus, the lower the Kappa (K) value the higher the diversity. Once the Kappa for each tree is computed, the Kappa Pruning algorithm sorts them in increasing order of K and selects the first M trees (user defined) for the subforest. Similarly, Ruta and Gabrys [40] proposed several forest pruning techniques by combining different search methods and selection criteria. One notable heuristic-based technique is "Minimum Individual Error Pruning" which computes the individual accuracy of each tree and then sorts them in non-increasing order. Then, the first M trees are selected for the subforest. The main problem of these heuristic-based approaches is that the size of subforest needs to be determined by the user which may lead to a sub-optimal subforest.

There are some forest pruning techniques based on general hill climbing strategy; in particular "Reduce-Error Pruning" [30,33,40]. In this technique, at first the most accurate tree is selected for the subforest. Then trees are added to the subforest one by one in such a way that the ensemble accuracy on the training data set increases from each addition. The addition of the u-th tree can be expressed by Eq. (2) where k is the number of trees that have not been included in a forest of size $u-1$.

$$\arg\max_{k} \sum_{i=1}^{N_{Train}} \left(h_k(\mathbf{R}_i) + \sum_{t=1}^{u-1} h_{s_t}(\mathbf{R}_i) \right) = y_i \tag{2}$$

Reduce-Error pruning was found very effective in literature [40]. Similar techniques such as "Complementariness Measure" [33] grows the subforest from the most accurate tree. Then at each iteration, the tree having the highest disagreement (diversity) with the current subforest is included according to the following (Eq. (3)).

$$\arg\max_{k} \sum_{i=1}^{N_{Train}} I\left(h_k(\mathbf{R}_i) = y_i \oplus \sum_{t=1}^{u-1} h_{s_t}(\mathbf{R}_i) = y_i \right) \tag{3}$$

Some of the notable forest pruning algorithms based on hill climbing strategy include "Margin Distance Minimization" [33] and more improved "Orientation Ordering" [32,34]. However, a recent heuristic-based technique called "Ensemble Pruning via Individual Contribution Ordering (EPIC)" has shown to be more effective than Orientation Ordering [29]. The key idea of EPIC is based on determining the contribution of individual trees in a forest in accordance with the following cases [36]:

(I) The tree predicts a record correctly and in a minority group in the forest it belongs to. That is, the forest predicts wrongly for the record.
(II) The tree predicts correctly and in the majority group in the forest it belongs to. That is, the forest also predicts correctly.
(III) The tree predicts incorrectly and in the minority group in the forest it belongs to.
(IV) The tree predicts incorrectly and in the majority group in the forest it belongs to.

Although EPIC is based on above mentioned four cases, they differ from Partalas et al. [36] as the latter did not draw any comparative relationship of the four cases for subforest selection. In EPIC [29], the authors concluded that forest members with correct predictions make positive contributions and incorrect predictions make negative contributions. To be more precise, correct predictions that are in the minority group bear more positive contributions than correct predictions from the majority group. Conversely, incorrect predictions that are in the minority group bear less negative contributions than incorrect predictions from the majority group. However, EPIC is not able to distinguish among intra-case differences and thus fails to generate reasonable contributions for many cases.

3. Motivation and problem statement

Despite the fact that both individual accuracy and diversity are instrumental to determine high quality individual trees in the context of a forest; yet till date no data set (problem) independent relationship exists between individual accuracy and diversity for optimum forest performance [41,48]. However, they are independently comparable among themselves.

For example, we can find the set of trees T^{Acc} that are more (or equally) accurate than the average of individual accuracy A of a forest $T = \{T_1, T_2, \ldots, T_z\}$ as follows, where t_i^{Acc} is the accuracy of T_i.

$$T^{Acc} = \{T_i : t_i^{Acc} \geq A\} | A = \frac{1}{|T|} \sum_{j=1}^{|T|} t_j^{Acc} \tag{4}$$

Similarly, we find the set of trees T^{Div} that are more (or equally) diverse than the average diversity K of all trees in the forest as follows, where t_i^K is the diversity (Kappa) of T_i. In this study, we use the Kappa value (K) as a measure of diversity as was done before.

$$T^{Div} = \{T_i : t_i^k \leq K\} | K = \frac{1}{|T|} \sum_{j=1}^{|T|} t_j^K \tag{5}$$

By applying both Eqs. (4) and (5), we are able to recognize those high quality trees that are simultaneously more (or equally) accurate and diverse than the averages in a forest. Now the question is, "*How these high quality trees perform as a subforest*"? In search of the answer, we conduct an elaborate experimentation involving twenty (20) well known data sets that are publicly available from the UCI Machine Learning Repository [26] that are described in Table 1. For example, the Chess data set has 36 non class attributes, 3196 records with two (02) distinct class values.

For our experimentation, we remove records with missing values (Table 1 shows the number of records with no missing values) and identifier attributes such as *Transaction_ID* from each applicable data set. We generate 100 trees for every decision forest since the number is considered to be large enough to ensure convergence of the ensemble effect [2,6,14]. We use Gini Index [10] as the splitting criteria, as suggested in Random Forest [9]. The minimum Gini Index value is set to 0.01 for any attribute to qualify for splitting a node of a tree. Each leaf node of a tree contains at least two records and no further post-pruning is applied in order to keep the results unbiased from the pruning methods. We apply majority voting to aggregate results for forest classification [9,37].

The experimentation is conducted by a single machine with Intel(R) 3.4 GHz processor and 8 GB Main Memory (RAM) running under 64-bit Windows 7 Enterprise Operating System. All the results reported in this paper are obtained using 10-fold-cross-validation (10-CV) [3,24,25] for every data set. In 10-CV, a data set is first randomly divided into 10 horizontal segments/partitions. The segments are mutually exclusive meaning that they do not have any overlapping records. Each segment in turn is considered to be the testing data set (out of bag samples) while for the same turn the remaining nine segments are considered to be the training data set. Thus, we get 10 training data sets and 10 corresponding testing data sets. We do not divide training data sets further to generate validation data sets as it is found that the performances of ensembles and pruned ensembles are better when generated from larger training data sets [32].

In our experimentation, we generate 100 trees from each training data set (thus 1000 trees in total) for each decision forest algo-

Table 1
Description of the data sets.

Data set name (DS)	Non-class attributes	Records	Distinct class values
Balance Scale (BS)	04	625	3
Breast Cancer (BC)	33	602	2
Car Evaluation (CE)	06	1728	4
Chess (CHS)	36	3196	2
Credit Approval (CA)	15	653	2
Dermatology (DER)	34	358	6
Ecoli (EC)	07	336	8
Glass Identification (GI)	09	214	6
Hayes-Roth (HR)	04	132	3
Hepatitis (HEP)	19	80	2
Image Segmentation (IS)	19	2310	7
Ionosphere (ION)	34	351	2
Iris (IRS)	04	150	3
Libras Movement (LM)	90	360	15
Liver Disorder (LD)	06	345	2
Pima Indians Diabetes (PID)	08	768	2
Sonar (SON)	60	208	2
Statlog Heart (SH)	13	270	2
Statlog Vehicle (SV)	18	846	4
Thyroid-New (TN)	05	215	3

rithm and then evaluate their performance with the corresponding testing data sets. All the performance indicators reported in this paper are the average values obtained from the 10 testing data sets and the best results are stressed through **bold-face**.

In literature, subforests are generated particularly from Bagging ensembles citing it's robustness [7,29,33–35,48]. In our study, we generate subforests of high quality trees from both Bagging and state-of-the-art Random Forest ensembles in the following ways:

(a) Sub_A: The subforest includes trees that are more or equally accurate than the average of individual accuracies (A) of the original forest (Applying Eq. (4)).
(b) Sub_D: The subforest includes trees that are more or equally diverse than the average diversity (D) of the original forest (Applying Eq. (5)). Note that in Eq. (5) K has been used as a measure of D and thus the lower the K, the higher the D.
(c) $Sub_{A.D}$: The subforest includes trees that are simultaneously more or equally accurate and diverse than their averages of the original forest ($Sub_{A.D} = Sub_A \cap Sub_D$).
(d) Sub_{All}: The subforest that includes all trees of the original forest.

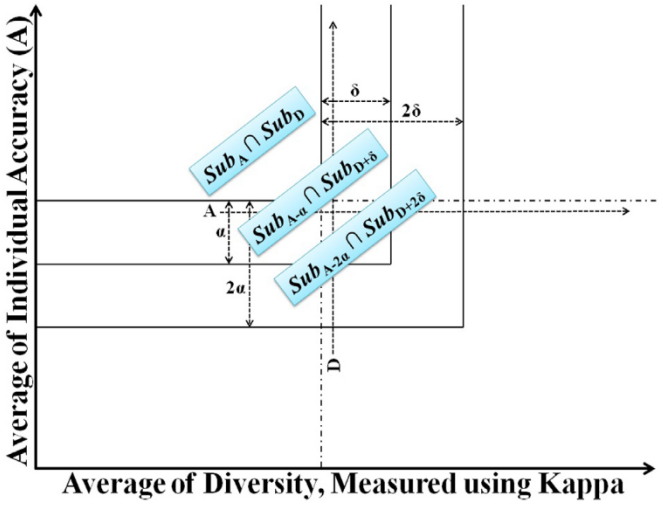

Fig. 1. Subforest extension.

Ensemble Accuracy (EA) and the Ensemble Size (ES) are two of the most important performance indicators for any subforest selection technique as they intend to reduce ES as much as possible while retaining or increasing EA. We now present EA (in percentage) and ES (in number of trees) of the subforests from Bagging and Random Forest ensembles in Tables 2 and 3 for all data sets considered. We see from the presented results that the subforests (Sub_A, Sub_D and $Sub_{A.D}$) built from both Bagging and Random Forest are inferior to the original forest in terms of EA. More importantly, $Sub_{A.D}$ performed worst among all and this is due to very low number of trees appeared as high quality trees which may not be enough to ensure convergence of the ensemble effect for many of these data sets [6,14].

To extend the number of high quality trees with retaining the notion that high quality trees are simultaneously more accurate as well as more diverse than others, we first calculate the standard deviation of individual accuracies (α) and diversities (δ) of the original forest. Then, we generate $Sub_{A-\alpha}$ which extends the trees of Sub_A by including trees that are more or equally accurate than $A-\alpha$ of the original forest. Thus, $Sub_{A-\alpha}$ includes trees that are less accurate than A but the range is limited by α. In the same way, we generate $Sub_{D+\delta}$ which extends the trees of Sub_D by including trees that are more or equally diverse than $D+\delta$. Thus, $Sub_{D+\delta}$ includes some trees that are less diverse than D but the range is limited by δ. We then generate $Sub_{A-\alpha.D+\delta} = Sub_{A-\alpha} \cup Sub_{D+\delta}$ and accordingly $Sub_{A-2\alpha.D+2\delta} = Sub_{A-2\alpha} \cup Sub_{D+2\delta}$ as the extensions of $Sub_{A.D}$. This extension can be further illustrated by Fig. 1.

To compare the relative performance (EA) and size (ES) of the subforests defined so far, we present the averages of EA and ES for all data sets for Bagging and Random Forest in Table 4. From Table 4 we see that ES progressively increases from $Sub_{A.D} \longrightarrow Sub_{A-\alpha.D+\delta} \longrightarrow Sub_{A-2\alpha.D+2\delta} \longrightarrow Sub_{All}$; however EA does not follow the same trend. For Bagging EA is the highest for $Sub_{A-2\alpha.D+2\delta}$ whereas for Random Forest the highest EA goes to $Sub_{A-\alpha.D+\delta}$. However, it is important to note that both $Sub_{A-\alpha.D+\delta}$ and $Sub_{A-2\alpha.D+2\delta}$ perform better than Sub_A, Sub_D and Sub_{All} for both Bagging and Random Forest. We already know, both individual accuracy and diversity contribute to increase in EA. This phenomenon is reflected in the results presented in Table 4 where both $Sub_{A-\alpha.D+\delta}$ and $Sub_{A-2\alpha.D+2\delta}$ perform better than Sub_A and Sub_D. Furthermore, similar to the findings of literature [29,30,32,33,36,53], both $Sub_{A-\alpha.D+\delta}$ and $Sub_{A-2\alpha.D+2\delta}$ perform better in EA than the complete forest Sub_{All} as they can exclude some trees that are simultaneously less accurate as well as less di-

Table 2
EA and ES of Sub_A, Sub_D, $Sub_{A.D}$ and Sub_{All} from Bagging.

DS	EA				ES			
	Sub_A	Sub_D	$Sub_{A.D}$	Sub_{All}	Sub_A	Sub_D	$Sub_{A.D}$	Sub_{All}
BS	76.67	82.09	74.24	77.48	50.60	49.40	10.10	100.00
BC	80.98	78.88	78.01	80.98	52.60	46.50	11.30	100.00
CE	93.86	93.86	93.00	93.34	53.00	47.80	9.40	100.00
CHS	98.78	98.72	98.97	97.87	47.80	48.40	34.70	100.00
CA	86.83	86.38	86.37	86.37	28.70	28.90	19.70	100.00
DER	90.78	50.31	39.73	88.50	64.80	31.80	0.90	100.00
EC	83.10	83.96	77.30	83.10	56.60	31.00	11.20	100.00
GI	72.69	71.81	64.50	74.12	61.40	38.70	16.70	100.00
HR	66.67	75.59	61.85	71.28	49.00	50.30	15.00	100.00
HEP	85.00	83.75	81.25	85.00	58.60	43.40	11.00	100.00
IS	92.68	89.83	89.44	92.64	71.50	46.70	33.00	100.00
ION	91.74	92.02	91.17	92.59	52.80	48.60	13.80	100.00
IRS	96.00	93.33	80.67	95.33	51.70	27.90	5.30	100.00
LM	75.28	73.33	61.11	76.95	51.40	48.10	4.40	100.00
LD	69.79	68.87	69.24	68.65	58.70	42.90	17.90	100.00
PID	75.45	74.12	74.13	75.58	57.30	41.10	19.50	100.00
SON	78.50	79.86	37.00	80.71	50.80	49.10	2.30	100.00
SH	81.48	79.63	77.78	81.11	54.10	46.20	10.80	100.00
SV	73.55	74.73	71.87	73.56	58.70	43.70	11.90	100.00
TN	93.61	94.56	93.13	93.61	55.00	45.90	10.50	100.00
Avg	83.17	81.28	75.04	**83.44**	54.26	42.82	13.47	100.00

Table 3
EA and ES of Sub_A, Sub_D, $Sub_{A.D}$ and Sub_{All} from Random Forest.

DS	EA				ES			
	Sub_A	Sub_D	$Sub_{A.D}$	Sub_{All}	Sub_A	Sub_D	$Sub_{A.D}$	Sub_{All}
BS	79.24	84.33	77.75	80.50	51.90	49.20	9.10	100.00
BC	77.83	77.30	75.81	77.92	55.80	46.30	12.30	100.00
CE	90.96	83.29	87.44	91.19	84.90	29.70	14.70	100.00
CHS	94.84	83.97	82.16	95.22	72.40	42.90	15.30	100.00
CA	86.68	81.16	83.76	86.07	76.80	40.90	18.50	100.00
DER	87.82	62.03	73.37	86.96	64.40	37.20	4.50	100.00
EC	84.62	85.43	85.43	84.97	60.00	38.20	13.60	100.00
GI	74.12	69.83	68.95	74.12	59.90	40.80	15.10	100.00
HR	72.62	73.39	63.39	69.54	49.30	49.30	13.10	100.00
HEP	86.25	85.00	81.25	86.25	71.70	43.40	17.80	100.00
IS	97.19	97.23	95.80	97.14	70.60	27.60	15.90	100.00
ION	94.02	93.45	93.17	93.73	52.50	47.50	12.90	100.00
IRS	95.33	93.33	84.00	96.00	54.20	26.90	9.50	100.00
LM	74.72	75.00	51.11	76.11	50.20	49.50	2.40	100.00
LD	68.40	68.36	66.63	71.48	62.30	42.70	20.20	100.00
PID	76.22	75.83	75.38	75.95	60.20	41.90	22.50	100.00
SON	80.86	81.07	27.00	83.07	52.10	48.20	1.30	100.00
SH	82.96	82.97	80.37	82.96	51.40	47.00	10.70	100.00
SV	74.27	74.02	73.05	74.14	59.70	42.90	13.30	100.00
TN	95.04	95.99	93.90	94.56	60.50	38.70	8.30	100.00
Avg	83.70	81.15	75.99	**83.89**	61.04	41.54	12.55	100.00

Table 4
Comparative analysis among Sub_A, Sub_D, $Sub_{A.D}$, $Sub_{A-\alpha.D+\delta}$, $Sub_{A-2\alpha.D+2\delta}$ and Sub_{All}.

Subforests	Bagging		Random Forest	
	EA	ES	EA	ES
Sub_A	83.17	54.26	83.70	61.04
Sub_D	81.28	42.82	81.15	41.54
$Sub_{A.D}$	75.04	13.47	75.99	12.55
$Sub_{A-\alpha.D+\delta}$	83.40	74.08	**84.06**	78.20
$Sub_{A-2\alpha.D+2\delta}$	**83.60**	96.49	83.91	95.18
Sub_{All}	83.44	100.00	83.89	100.00

verse than a bulk majority of trees (seen as relatively detrimental trees in literature [29,32,36]).

Another very interesting finding related to the increase of ES is that both Bagging and Random Forest ensemble follow Bell-Shaped distribution. On an average $Sub_{A.D}$ contains 13.01% trees (13.47% from Bagging and 12.55% from Random Forest), $Sub_{A-\alpha.D+\delta}$ contains 76.14% trees (74.08% from Bagging and 78.20% from Random Forest) and $Sub_{A-2\alpha.D+2\delta}$ contains 95.84% trees (96.49% from Bagging and 95.18% from Random Forest) (see Table 4). This implies that there are limited number of high quality (13.01%) and low quality (4.16%) trees with bulk majority of average-quality trees with respect to Bagging and Random Forest ensemble. Although, both $Sub_{A-\alpha.D+\delta}$ and $Sub_{A-2\alpha.D+2\delta}$ perform fairly similar in terms of EA, there is a considerable gap between them in terms number of trees. Hence, inquisitively we search for better subforests in between $Sub_{A-\alpha.D+\delta}$ and $Sub_{A-2\alpha.D+2\delta}$ by slowly decreasing α and increasing δ to form $Sub_{A-1.1\alpha.D+1.1\delta} = Int_1$, $Sub_{A-1.2\alpha.D+1.2\delta} = Int_2$ and so on. Figs. 2 and 3 show the changes of EA from $Sub_{A-\alpha.D+\delta}$ to $Sub_{A-2\alpha.D+2\delta}$ through intermediate subforests for every data sets considered for Bagging and Random Forest respectively. The vertical scales of Figs. 2 and 3 indicate EA and the horizontal scales present the averages of ES for all data sets. From Figs. 2 and 3 we find that there is no data set (problem) independent size of opti-

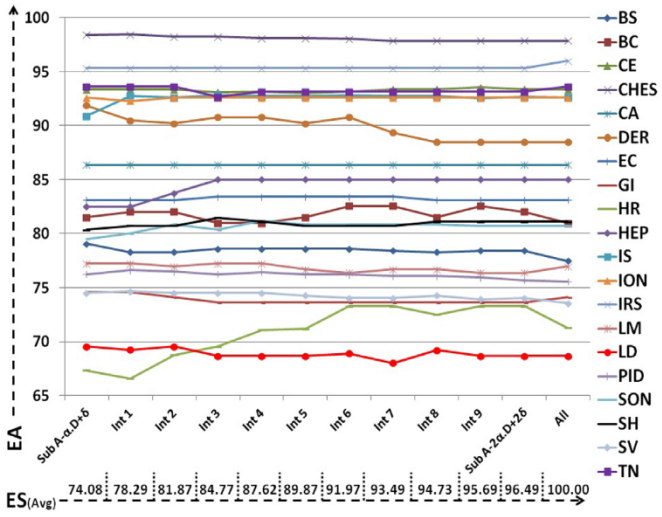

Fig. 2. EA of intermediate subforests for Bagging.

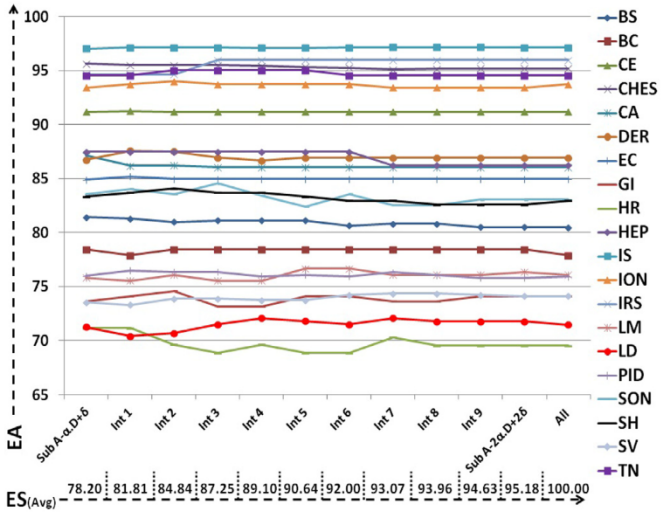

Fig. 3. EA of intermediate subforests for Random Forest.

mal subforest for any of the two ensembles. For one data set EA may increase with the inclusion some trees while for another data set similar act may decrease EA. For example, for Bagging ensemble (see Fig. 2) the EA for Breast Cancer (BC) data set decreases from Int_7 to Int_8 (technically Int_8 is said to have more trees than Int_7) whereas for Liver Disorder (LD) data set the EA increases from Int_7 to Int_8. Thus the question of finding problem independent **optimal subforest** remains to be solved by GA with the aid of relatively good-quality trees as initial population.

4. The proposed technique

The main contribution of the proposed forest pruning (i.e. subforest selection) technique is infusing high quality trees as the initial population for GA. To the best of our knowledge, no previous techniques so far have dealt with high quality initial population selection in any form for GA-based subforest selection.

Genetic Algorithm (GA in this paper) is a class of computational framework inspired by evolution [46]. GA was first introduced by John H. Holland as an adaptation of natural evolution (***survival for the fittest***) in computing [18]. GA encodes a potential solution in a simple data structure called chromosome. Typically, the execution of a GA begins with a population of randomly defined chromosomes. Then GA moves forward by applying genetics-inspired operators such as crossover and mutation to create a new population. Then these chromosomes are evaluated routinely and chromosomes representing better solution remain in the process to be given more chance for reproduction. The chromosome with the best solution so far is reported as the output of GA. We follow this general structure of GA blended with our novel contribution which is the initial population selection and present them as follows.

4.1. Initial population selection

Population is a set of chromosomes. Thus, *Initial Population Selection* is materialized from the encoding of the initial chromosomes. The first component of initial population selection is chromosome encoding.

Chromosome encoding: We already know, a chromosome is a potential solution encoded in a data structure. Hence, in our case a chromosome represents a subforest. Let there are T number of trees in a forest (As we generate 100 trees in a forest, $T = 100$ in our case). To encode a chromosome, we use a string of T binary digits where i-th tree is assigned to i-th digit. A digit has one of two values 1 or 0, meaning *selected* or *not selected* for the respective tree. For example, a chromosome $Cr = 1001000101$ means that out of 10 trees 1st, 4th, 8th and 10th tree (four in total) are selected to constitute the subforest.

In literature, the number of trees selected for subforest (M) is chosen randomly [20,40]. Now the question is how these M among T will be selected? In literature, they are sampled randomly [20,40]. We have noticed that a subforest generated from good-quality trees performed better than other subforests with similar/comparable size (on an average $Sub_{A-\alpha.D+\delta}$ performed better than Sub_A and Sub_D for both Bagging and Random Forest ensembles, see Table 4). Thus, it is reasonable that the optimal subforest (the subforest GA is looking for) require relatively good-quality trees as it's member. However, selecting trees randomly for a subforest may not guaranty the presence of sufficient number of good-quality trees making it difficult to transform itself into optimal/near optimal subforest in a reasonable number of iterations as many other components of GA such as crossover and mutation are also randomly driven. On the other hand, only the high quality trees do not constitute the optimal subforest as we have already seen that $Sub_{A\cap D}$ could not perform well. Though $Sub_{A-\alpha.D+\delta}$ and $Sub_{A-2\alpha.D+2\delta}$ performed relatively well, prioritizing only good-quality trees may narrow the search space for GA. Hence, we intend to keep a balance between selecting good-quality trees and randomization for chromosome encoding.

In our proposed technique, we encode 20 chromosomes to constitute the population ($|\mathbf{P}| = 20$) as was done in literature [20]. In the population, each odd chromosome (10 in total) focuses on accumulating relatively good-quality trees and each even chromosome (10 in total) accumulates trees randomly as was done in literature [20,40]. Accumulation of relatively good-quality trees in a chromosome is done as follows:

In order to prioritize relatively good-quality trees in odd chromosomes we apply stratified sampling instead of simple random sampling. Stratified sampling is a probability sampling procedure in which the target set of items is first separated into mutually exclusive, homogeneous segments called **strata** and then the items are selected proportionately or disproportionately to the size of strata [11]. In our case, we first define three stratum as follows:

(I) **Stratum 1** (S_1): S_1 includes the trees of $Sub_{A.D}$.
(II) **Stratum 2** (S_2): S_2 includes the trees of $Sub_{A-\alpha.D+\delta} \setminus Sub_{A.D}$.

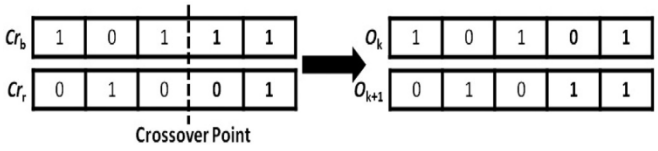

Fig. 4. Crossover operation.

(III) **Stratum 3** (S_3): S_3 includes the trees of $(Sub_{A-2\alpha.D+2\delta} \backslash Sub_{A-\alpha.D+\delta})$.

On an average, S_1, S_2 and S_3 altogether cover 96% trees for Bagging and Random Forest as they follow Bell-Shaped distribution. The rest 4% low quality trees are excluded from these strata. After the strata are defined, M is selected randomly between 1 to 100. However, if $M > |S_1| + |S_2| + |S_3|$, M is set to $|S_1| + |S_2| + |S_3|$. We then collect M trees from S_1, S_2 and S_3 using Disproportionate Stratified Sampling (DSS). Our form of DSS gives absolute priority of S_1 over S_2 and S_2 over S_3. For example, if $M = |S_1|$ all trees for the chromosome are selected from S_1. But when $M \leq |S_1|$, M trees are randomly selected from S_1. However, when $M > |S_1|$ and $M < |S_1| + |S_2|$ all trees from S_1 are first selected. Then the rest $M - S_1$ are selected randomly from S_2. In the same way as before, when $M = |S_1| + |S_2|$ all trees of $S_1 \cup S_2$ are selected for the chromosome. The selection process remains the same when $M > |S_1| + |S_2|$.

For even chromosomes, M is selected randomly between 1 to 100. Then all M trees are randomly selected from T (which may include some of the 4% low quality trees) as was done in literature [20,40]. In this way, when all 20 chromosomes are generated the initial population is designated as the current population ($\mathbf{P}_{Curr}$). After the generation of $\mathbf{P}_{Curr}$, the best chromosome is stored as the so far best chromosome Cr_{SFBest} (see Step 1 of Algorithm 1).

4.2. Crossover and Mutation

Crossover is a fundamental component of GA which is critical for the evolution of new population [38,46]. Generally, crossover operation is applied on a chromosome pair (parents) where they swap segments to form the offspring. In our proposed technique, we select the best chromosome of $\mathbf{P}_{Curr}$ (Cr_b) as the first chromosome of a pair. To select the second chromosome of the pair, we use roulette wheel technique [27,38] where a chromosome Cr_r ($\neq Cr_b$) is selected with a probability $p(Cr_r) = \frac{EA(Cr_r)}{\sum_{i=1}^{|\mathbf{P}_{Curr}|} EA(Cr_i)}$ (where $EA(Cr_r)$ is the ensemble accuracy of the chromosome/subforest Cr_r) affirming better chromosomes have greater chance of selection over weaker ones. Once a chromosome pair is selected, they are excluded from the process of choosing the upcoming pairs; all pairs are chosen in the same process as described. The motivation behind roulette wheel selection is to induce some randomness in choosing compatible peer for the best available chromosome of a population (see Step 2 of Algorithm 1).

After pairing the chromosomes, standard 1-point crossover operation is applied on them as described in Fig. 4. A single crossover point is selected randomly between 1 and T for each parent pair, where T is the total number of trees in the complete forest and hence the full length of each chromosome. During the crossover operation parent chromosomes swap genes; left genes (i.e. genes in the left side of the crossover point) of one chromosome join the right genes of another chromosome. After crossover operation all the parent pairs are converted into offspring pairs with same number of bits.

We then perform 1-bit flipping mutation on each offspring in the following way: We first select a bit randomly between 1 and T for an offspring. If the randomly selected bit is 0 then we flip it to 1 and vice versa (see Step 3 of Algorithm 1). In this way, mutation helps to induce slight randomness in search directions.

Algorithm 1:

input: A Decision Forest DF.
output: A Subforest Sub_{PGA}.

Required:
  $T \longleftarrow$ 100 (Size of DF) ; $|\mathbf{P}| \longleftarrow$ 20 (Size of Population);$J \longleftarrow$ 20 (Number of Iterations);
end
$\mathbf{P}_{Curr} \leftarrow \emptyset$;**Step 1: Initial Population Selection**
  for $(i = 1$ **to** $|\boldsymbol{P}|)$ **do**
    if $(i \mod 2 = 0)$ **then** $Cr_i \leftarrow$ Get_Cr_Random();
    else $Cr_i \leftarrow$ Get_Cr_Strata();
    $\mathbf{P}_{Curr} \leftarrow \mathbf{P}_{Curr} \cup \{Cr_i\}$;
  end
  $Cr_{SFBest} \leftarrow$ Get_Best_Cr($\mathbf{P}_{Curr}$);
end
for $(j = 1$ **to** $J)$ **do**
  $Cr_{CurrBest} \leftarrow$ Get_Best_Cr($\mathbf{P}_{Curr}$);
  Step 2: Crossover and Elitist Operation
    $\mathbf{P}_{TempCurr} \leftarrow \mathbf{P}_{Curr}$;
    for $(k = 1$ **to** $|\boldsymbol{P}_{Curr}|/2)$ **do**
      $Cr_b \leftarrow$ Get_Best_Cr($\mathbf{P}_{TempCurr}$);
      $Cr_r \leftarrow$ Get_Roulette_Cr($\mathbf{P}_{TempCurr}$);
      /* $Cr_b \neq Cr_r$ */
      $\mathbf{O} \leftarrow$ Crossover(Cr_b, Cr_r) ; /* Two Offspring $\mathbf{O} \leftarrow \{O_k, O_{k+1}\}$ are generated */
      $\mathbf{P}_{TempCurr} \leftarrow \mathbf{P}_{TempCurr} \cap \{Cr_b\} \cap \{Cr_r\}$;
      $\mathbf{P}_{Mod} \leftarrow \mathbf{P}_{Mod} \cup \{O_k\} \cup \{O_{k+1}\}$;
    end
    Elitist_Operation(Cr_{SFBest}, $Cr_{CurrBest}$, $\boldsymbol{P}_{Mod}$);
  end
  Step 3: Mutation and Elitist Operation
    for $(k = 1$ **to** $|\boldsymbol{P}_{Mod}|)$ **do**
      $t \leftarrow$ Gen_Rnd_Num(1, T) ;
      /* Random Number between 1 and T */
      $\mathbf{P}_{Mod} \leftarrow$ 1-Bit_Flipper(Cr^k_{Mod}, t);
    end
    Elitist_Operation(Cr_{SFBest}, $Cr_{CurrBest}$, $\boldsymbol{P}_{Mod}$);
  end
  Step 4: Chromosome Selection for the Next Iteration
    $\mathbf{P}_{Pool} \leftarrow \mathbf{P}_{Curr} \cup \mathbf{P}_{Mod}$;
    $\mathbf{P}_{Curr} \leftarrow$ Get_Roulette_Pop($\mathbf{P}_{Pool}$, J);
  end
end
Step 5: Rectification of So Far Best Chromosome
  for $(k = 1$ **to** $T)$ **do**
    if $(Cr^k_{SFBest} == 0)$ **then** Rectify(Cr'_{SFBest}, k));
  end
  for $(k = 1$ **to** $T)$ **do**
    if $(Cr^k_{SFBest} == 1)$ **then** Rectify(Cr'_{SFBest}, k));
  end
end
$Sub \leftarrow Cr_{SFBest}$;
return Sub_{PGA};

4.3. Elitist operation

The elitist operation searches for a new Cr_{SFBest} after both crossover and mutation operations as they bring changes in $\mathbf{P}_{Curr}$ [38]. At the beginning of an iteration, all the chromosomes

Algorithm 2:

```
Elitist_Operation(Cr_SFBest, Cr_CurrBest, P_Mod) ;
begin
    Cr_ModBest ← Get_Best_Cr(P_Mod) ;
    if Cr_ModBest > Cr_SFBest then
        Cr_SFBest ← Cr_ModBest ;
    end
    Cr_ModWorst ← Get_Worst_Cr(P_Mod) ;
    if Cr_CurrBest > Cr_ModWorst then
        Cr_ModWorst ← Cr_CurrBest ;
    end
end
```

Algorithm 3:

```
Rectify(Cr_SFBest, k) ;
begin
    Cr'_SFBest ← 1-Bit_Flipper(Cr^k_SFBest, k) ;
    if Cr'_SFBest > Cr_SFBest then
        Cr_SFBest ← Cr'_SFBest ;
    end
end
```

of $\mathbf{P}_{Curr}$ are duplicated in $\mathbf{P}_{TempCurr}$. Also, the best chromosome of $\mathbf{P}_{Curr}$ is stored as $Cr_{CurrBest}$. Then, at first crossover is applied on $\mathbf{P}_{TempCurr}$ to form $\mathbf{P}_{Mod}$.

Next, the elitist operation compares $Cr_{ModBest}$ (i.e. the best chromosome in $\mathbf{P}_{Mod}$) with Cr_{SFBest}, and if $Cr_{ModBest}$ is better than Cr_{SFBest} we replace Cr_{SFBest} with $Cr_{ModBest}$. Also, we replace the worst chromosome in $\mathbf{P}_{Mod}$ with $Cr_{CurrBest}$ if $Cr_{CurrBest}$ is better than that chromosome (see Algorithm 2) and then $Cr_{CurrBest}$ is recomputed from $\mathbf{P}_{Mod}$. In this way, a competent $Cr_{CurrBest}$ is always retained in any $\mathbf{P}_{Mod}$ so that it can always appear in the forthcoming modifications and does not loose any chance of improvement. $\mathbf{P}_{Mod}$ is further modified by mutation and then again the elitist operation is applied in the same way as before.

4.4. Chromosome selection for the next iteration

At the end of each iteration, we check whether the modifications done by the genetic operations are counter-productive. By a counter productive modification we mean that most of the chromosomes in a population ($\mathbf{P}_{Mod}$) at the end of an iteration (after applying Crossover, Mutation and Elitist operations) are inferior to the chromosomes of $\mathbf{P}_{Curr}$, which is the input population to the iteration. Such a counter productive modification indicates that $\mathbf{P}_{Mod}$ is inferior to $\mathbf{P}_{Curr}$. Note that, the crossover and mutation operations are applied on $\mathbf{P}_{TempCurr}$ and thus $\mathbf{P}_{Curr}$ is preserved unchanged during an iteration.

If $\mathbf{P}_{Mod}$ is inferior to $\mathbf{P}_{Curr}$ and if $\mathbf{P}_{Mod}$ becomes $\mathbf{P}_{Curr}$ for the next iteration then there is strong possibility of continuous degradation in subsequent iterations. This may promote search in a wrong direction of the solution space.

To prevent this degrading scenario, at the end of each iteration we create a pool of chromosomes ($\mathbf{P}_{Pool}$) by adding chromosomes of $\mathbf{P}_{Curr}$ and then $\mathbf{P}_{Mod}$. Hence, $\mathbf{P}_{Pool}$ consists of 40 chromosomes. We then apply roulette wheel technique to select 20 chromosomes from the 40-chromosomes of $\mathbf{P}_{Pool}$. These 20 selected chromosomes then form the new $\mathbf{P}_{Curr}$ for the next iteration (see Step 3 of Algorithm 1). This encourages that good chromosomes are used in the next iteration.

4.5. Rectification of so far best chromosome

GAs are able to dodge local optima and traverse a wide range of search space with the aid of iterative crossover and mutation. However, the potential solution are randomly driven and thus can be fine-tuned in the direction of optimal/near-optimal solution. In literature, Kim et al. [20] employed two types of local search operations namely Sequential Search Operations (SSO) and Combinational Search Operations (CSO) on Cr_{SFBest} at the end of all iterations. As CSO is more computationally expensive, we employ SSO for rectification of Cr_{SFBest} in the following form:

In the first phase, each bit with a value of 1 in Cr_{SFBest} is flipped to 0 (i.e. each tree is excluded from the so far best subforest) one by one. After each bit change from 1 to 0, the EA is checked for the chromosome. If the exclusion does not increase the EA the bit value is reverted from 0 to 1; otherwise the change persists. Similarly in the second phase, each bit with a value of 0 in Cr_{SFBest} is flipped to 1 (i.e. each tree is included in the so far best subforest) one by one. After each bit change from 0 to 1, the EA is checked for the chromosome. If the inclusion does not increase the EA the bit value is reverted from 1 to 0; otherwise the change persists. In this way, Cr_{SFBest} is rectified using SSO (see Step 5 of Algorithm 1 and Algorithm 3).

We now use the above mentioned operations to present the structure of the proposed technique, as follows. The proposed technique is also illustrated through Algorithm 1, and two sub-algorithms: Algorithm 2 and Algorithm 3.

Step 1. *Initial Population Selection*
for ($j = 1$ **to** J) **do**/* J is the number of iterations*/
Step 2. *Crossover and Elitist Operation*
Step 3. *Mutation and Elitist Operation*
Step 4. *Chromosome Selection for the Next Iteration*
End **for**
Step 5. *Rectification of So Far Best Chromosome*

5. Experimental validation

We now conduct an elaborate experimentation to evaluate the comparative performance of different subforest building algorithms. We use the same data sets that are shown in Table 1. Also, the experimental conditions remain the same as described in Section 3. We put together some of the most reputed subforest algorithms including Ordering Trees based on Individual Accuracy (Sub_{IA}) [40] and Individual Diversity (Sub_{ID}) [40], EPIC (Sub_{IC}) [29] and GA-based HGA (Sub_{HGA}) [20].

Sub_{HGA} can be seen as the predecessor of the proposed Sub_{PGA}. Sub_{HGA} was originally proposed to build sub-ensembles from a pool of 50 1-nearest neighbour classifiers [20]. In this study, we apply it to build a subforest in order to compare it with the proposed technique. Note that Sub_{HGA} produces the initial population completely randomly, whereas the main contribution of the proposed Sub_{PGA} is the careful selection of initial population (see Section 4.1). For the other genetic operations of Sub_{HGA} we use Crossover, Mutation, Elitist, Chromosome Selection for the Next Iteration and Rectification of So Far Best Chromosome operations as explained in Sections 4.2–4.5, respectively. The original Sub_{HGA} does not use the operation called Chromosome Selection for the Next Iteration. However, it is evident from the literature [4] that the use of such chromosome selection operation (also known as the Chromosome Health Improvement operation) improves the overall performance. Therefore, we use the operation in Sub_{HGA} in order to make it more competitive with the proposed technique. Similarly, we use Rectification of So Far Best Chromosome operation (see Section 4.5) for both Sub_{HGA} and Sub_{PGA}.

Table 5
EA of Sub_{IA}, Sub_{ID} and Sub_{IC} from Bagging.

DS	Sub_{IA}^{40}	Sub_{IA}^{60}	Sub_{IA}^{80}	Sub_{ID}^{40}	Sub_{ID}^{60}	Sub_{ID}^{80}	Sub_{IC}^{40}	Sub_{IC}^{60}	Sub_{IC}^{80}
BS	75.05	77.31	77.46	81.61	81.15	79.40	77.65	78.44	77.48
BC	81.51	80.98	81.51	78.44	79.50	80.98	81.51	79.93	80.98
CE	93.74	93.57	93.34	93.51	93.16	93.05	92.81	93.39	93.22
CHS	99.13	98.69	98.25	98.69	98.22	97.88	97.84	98.00	97.87
CA	86.38	86.37	86.37	86.38	86.22	86.37	86.37	86.37	86.37
DER	92.73	92.21	90.79	54.60	74.60	89.64	90.50	90.21	91.36
EC	83.66	83.36	83.36	83.36	82.45	82.75	84.01	83.71	83.10
GI	74.19	72.69	74.59	73.32	74.59	75.07	72.21	73.16	72.69
HR	64.36	62.82	68.21	76.36	73.28	67.33	66.46	65.49	68.67
HEP	85.00	85.00	85.00	83.75	85.00	83.75	85.00	82.50	82.50
IS	92.64	92.90	92.73	89.31	90.65	90.61	92.60	92.99	92.77
ION	92.03	91.73	92.30	92.03	92.30	92.87	92.30	92.59	92.59
IRS	95.33	96.00	94.67	94.00	94.00	94.00	94.67	94.67	94.67
LM	75.56	75.56	76.39	73.61	75.28	76.11	76.11	75.83	75.56
LD	70.38	68.91	68.36	67.73	68.91	69.79	69.79	69.20	69.24
PID	75.67	75.58	75.98	74.39	74.54	75.71	76.09	75.96	76.76
SON	79.00	78.36	80.86	78.86	80.57	80.21	77.86	79.21	80.71
SH	80.37	81.48	81.11	79.63	79.26	79.63	81.11	80.74	81.48
SV	74.37	73.45	73.79	75.20	73.33	73.45	73.54	74.83	73.66
TN	93.13	93.13	93.13	94.56	94.56	93.61	93.13	93.61	94.09
Avg	83.21	83.00	**83.41**	81.47	82.58	83.11	83.08	83.04	83.29

Table 6
EA of Sub_{IA}, Sub_{ID} and Sub_{IC} from Random Forest.

DS	Sub_{IA}^{40}	Sub_{IA}^{60}	Sub_{IA}^{80}	Sub_{ID}^{40}	Sub_{ID}^{60}	Sub_{ID}^{80}	Sub_{IC}^{40}	Sub_{IC}^{60}	Sub_{IC}^{80}
BS	78.29	80.03	79.87	83.68	84.47	83.05	79.39	79.88	80.19
BC	78.35	77.83	78.44	77.30	78.35	77.39	78.44	78.97	77.92
CE	90.32	90.73	91.02	86.58	89.70	90.79	90.39	90.96	91.02
CHS	94.72	94.65	95.12	80.77	90.93	95.53	92.31	95.69	96.00
CA	86.37	86.83	86.22	81.31	84.07	86.70	85.31	86.22	86.37
DER	90.97	88.96	87.54	65.01	80.94	86.84	89.30	87.87	86.16
EC	84.62	84.01	84.92	85.18	84.31	84.92	84.31	84.06	84.66
GI	76.50	74.12	74.59	69.35	73.56	74.12	77.45	76.02	74.59
HR	70.31	68.00	71.08	70.98	73.49	69.64	64.72	67.80	72.51
HEP	85.00	85.00	86.25	85.00	86.25	87.50	85.00	87.50	86.25
IS	97.40	97.14	97.14	97.36	97.23	97.06	97.32	97.10	97.06
ION	94.02	93.73	93.45	93.45	93.45	93.45	94.02	94.02	93.73
IRS	96.00	94.67	95.33	94.67	95.33	94.67	96.00	96.00	96.00
LM	75.83	74.72	76.11	75.00	73.89	75.83	73.61	74.45	76.67
LD	69.42	68.91	70.93	68.87	70.67	71.77	70.34	71.89	72.40
PID	75.72	76.75	75.85	76.08	75.20	75.85	76.07	76.47	76.21
SON	80.86	81.57	83.79	80.21	83.07	82.57	77.07	80.29	80.93
SH	81.48	84.45	83.70	82.22	83.34	82.59	82.22	82.59	83.33
SV	74.63	74.27	74.15	74.14	73.56	72.96	74.84	73.64	74.02
TN	95.51	95.04	94.56	95.99	95.04	94.56	94.56	94.56	94.56
Avg	83.82	83.57	84.00	81.16	83.34	83.89	83.13	83.80	**84.03**

For Sub_{IA}, Sub_{ID} and Sub_{IC}, we consider to generate 40, 60 and 80-trees subforests from 100-tree forest as was done in the literature [29]. Note that our proposed algorithm generates the number of trees in a subforest automatically without requiring any user defined number of trees. We now present the EA of Sub_{IA}, Sub_{ID} and Sub_{IC} for Bagging in Table 5 and for Random Forest in Table 6.

From Tables 5 and 6 we find similar exposition of previously presented results in Tables 2 and 3 that there is no data set (problem) independent optimal subforest for any of the contending algorithms (Sub_{IA}, Sub_{ID} and Sub_{IC}). On an average, for Bagging Sub_{IA}^{80} (subforest with 80 trees) delivers the best EA. However, the performance is not always related to the number of trees in subforests as Sub_{IA}^{40} performs better than Sub_{IA}^{60}. Besides, for Random Forest Sub_{IC}^{80} performs the best in terms of average EA. Due to these inconsistent performance of greedy approaches, the justification of GA-based subforest algorithm is further established and we now present head-to-head comparison between two contending GA-based subforest algorithms Sub_{HGA} and Sub_{PGA} in Table 7.

From Table 7 we see that for Bagging Sub_{PGA} wins over Sub_{HGA} on 15 data sets excluding 4 draws out of 20 data sets in terms of EA. Also, Sub_{PGA} is smaller than Sub_{HGA} on 12 data sets for Bagging. Similarly for Random Forest, out of 20 data sets Sub_{PGA} wins over Sub_{HGA} on 16 data sets excluding one draw. Besides, Sub_{PGA} is smaller than Sub_{HGA} on 12 data sets for Random Forest. On an average, Sub_{PGA} has clear edges over Sub_{HGA} performing better in both EA and ES for both Bagging and Random Forest. The results presented in Table 7 shows the influence of selecting high quality individual trees for initial population in GA-based subforest selection. We now put together the performances (EA and ES) of all subforests tested so far on both Bagging and Random Forest ensembles in Table 8.

From the results presented in Table 8, we see that on an average Sub_{PGA} performs better not only in terms of EA but also in terms of ES compared to other subforests. For example, for Bagging Sub_{PGA} has the highest EA (**84.48**) with second lowest ES (second to $Sub_{A.D}$). Similarly, for Random Forest Sub_{PGA} has the highest EA (**84.76**) with comparatively lower ES. The results presented so far brings about the fact that on an average Sub_{PGA} performs better than other contending algorithms.

Table 7
Comparative performance of Sub_{HGA} and Sub_{PGA} on Bagging and Random Forest.

	Bagging				Random Forest			
	EA		ES		EA		ES	
DS	Sub_{HGA}	Sub_{PGA}	Sub_{HGA}	Sub_{PGA}	Sub_{HGA}	Sub_{PGA}	Sub_{HGA}	Sub_{PGA}
BS	78.28	**80.03**	**55.70**	58.40	**82.43**	82.24	63.90	**61.60**
BC	81.51	**82.13**	**44.20**	45.00	77.48	**78.44**	41.70	**36.10**
CE	93.68	**93.86**	44.60	**38.60**	91.08	**91.54**	51.60	**49.10**
CHS	98.88	**99.19**	**22.50**	30.20	97.09	**97.56**	**42.00**	42.50
CA	85.59	**86.68**	**12.70**	13.20	86.60	**87.30**	**47.20**	48.20
DER	93.02	**93.59**	**23.70**	28.40	93.25	**94.06**	**35.10**	40.40
EC	**83.10**	**83.10**	53.60	**45.70**	82.62	**84.36**	41.00	**33.10**
GI	74.59	**74.71**	44.10	**35.30**	73.64	**74.12**	48.30	**48.20**
HR	71.18	**74.26**	**53.10**	59.60	69.64	**73.49**	**60.30**	60.80
HEP	**85.00**	**85.00**	29.70	**22.10**	**87.50**	86.25	**30.70**	37.20
IS	96.84	**96.93**	**36.90**	38.40	97.25	**97.36**	46.10	**43.90**
ION	91.74	**92.31**	24.10	**17.20**	92.45	**93.16**	24.70	**21.60**
IRS	95.33	**96.00**	39.20	**38.10**	94.67	**96.00**	43.00	**27.30**
LM	76.67	**77.56**	57.50	**44.90**	75.62	**75.83**	61.60	**47.40**
LD	68.75	**69.53**	48.20	**46.60**	69.24	**71.52**	**46.50**	51.90
PID	**76.33**	76.21	48.90	**47.20**	74.78	**76.20**	**46.30**	49.20
SON	**79.36**	**79.36**	64.10	**52.10**	**83.21**	82.36	**58.00**	60.00
SH	**81.11**	**81.11**	43.90	**42.80**	81.85	**83.70**	41.40	**32.50**
SV	73.78	**74.03**	**57.60**	58.20	73.48	**74.25**	65.10	**61.30**
TN	93.61	**94.00**	38.00	**28.30**	**95.42**	**95.42**	39.60	**32.30**
Avg	83.92	**84.48**	42.12	**39.52**	83.96	**84.76**	46.71	**44.23**

Table 8
Comparative analysis among subforests defined so far.

	Bagging		Random Forest	
Subforests	EA	ES	EA	ES
Sub_A	83.17	54.26	83.70	61.04
Sub_D	81.28	42.82	81.15	41.54
$Sub_{A,D}$	75.04	**13.47**	75.99	**12.55**
$Sub_{A-\alpha.D+\delta}$	83.40	74.08	84.06	78.20
$Sub_{A-2\alpha.D+2\delta}$	83.60	96.49	83.91	95.18
Sub_{IA}^{40} [40]	83.21	40.00	83.82	40.00
Sub_{IA}^{60} [40]	83.00	60.00	83.57	60.00
Sub_{IA}^{80} [40]	83.41	80.00	84.00	80.00
Sub_{ID}^{40} [40]	81.47	40.00	81.16	40.00
Sub_{ID}^{60} [40]	82.58	60.00	83.34	60.00
Sub_{ID}^{80} [40]	83.11	80.00	83.89	80.00
Sub_{IC}^{40} [29]	83.08	40.00	83.13	40.00
Sub_{IC}^{60} [29]	83.04	60.00	83.80	60.00
Sub_{IC}^{80} [29]	83.29	80.00	84.03	80.00
Sub_{HGA} [20]	83.92	42.12	83.96	46.71
Sub_{PGA} (Proposed)	**84.48**	39.52	**84.76**	44.23
Sub_{All}	83.44	100.00	83.89	100.00

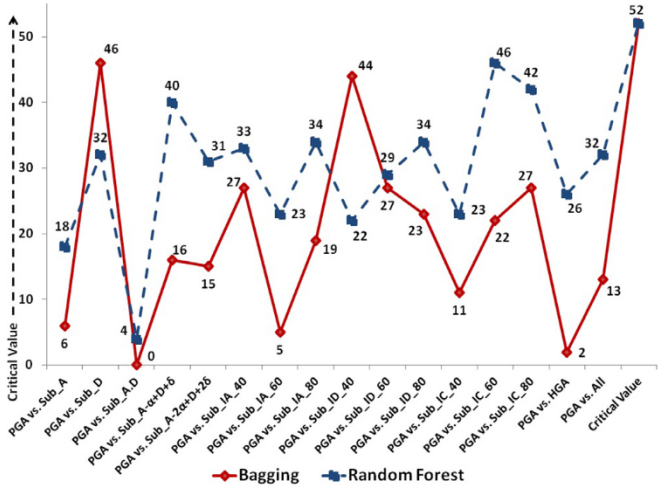

Fig. 5. Wilcoxon signed-ranks test on EA.

Now, to assess the statistical significance of the superiority of Sub_{PGA}, we conduct a statistical significance analysis called Wilcoxon Signed-Ranks Test [1,47]. We observe that EA results do not follow a normal distribution and thus do not satisfy the conditions for parametric tests. Hence, we perform a non-parametric one-tailed Wilcoxon Signed-Ranks Test [1,47] for $n = 20$ (number of data sets used) with the significance level $\alpha = 0.025$ (thus the critical value being 52) [31,45]. Wilcoxon Signed-Ranks Test is considered to be the preferable technique for comparing two classifiers [12]. Fig. 5 shows that Sub_{PGA} performs significantly better (in terms of EA) than all sixteen contending subforest algorithms (including Sub_{All}) over 20 widely used data sets as the test value remains lower than the critical value for every head-to-head comparisons for both Bagging and Random Forest ensembles. For example, for head-to-head comparison between Sub_{PGA} and Sub_{HGA} the test values are 2 and 26 for Bagging and Random Forest ensembles respectively (follow the solid line for Bagging and the dashed line for Random Forest in Fig. 5). This confirms that for both Bagging and Random Forest ensembles Sub_{PGA} performs significantly better than Sub_{HGA} as the test values remain much below than the critical value.

Now, we focus on the implication of Sub_{PGA} in a practical viewpoint. The proposed Sub_{PGA} not only increases EA for both Bagging and Random Forest ensembles but also shrinks the size of the forests substantially. On an average, for Bagging ensemble Sub_{PGA} prunes more than 60 trees to contain only 39.52 trees whereas for Random Forest ensemble Sub_{PGA} prunes more than 55 trees and accumulates only 44.23 trees. This reduction of size saves both Bagging and Random Forest ensembles from unnecessary memory and computational overhead.

Computational overhead for a decision forest mainly comprises of two components; first generating the decision forest and then predicting test records form the decision forest. Sub_{PGA} is searched from a fully grown decision forest through GA and thus it adds more computation in the process. At the same time, Sub_{PGA} is roughly of 50% of the original decision forest requiring 50% less voting overhead for predicting a single test record. Searching Sub_{PGA} from a fully grown decision forest mainly involves voting

different candidate subforests on the training data set through elitist operation. Hence the computational gain from Sub_{PGA} can be modelled based on how much voting it required to be searched and how much predictions it needs to perform to pay off the searching. For an approximate assessment of how much voting may require to search for Sub_{PGA} we consider a training data set with 1000 records. For implementing Sub_{PGA}, we use 20 chromosomes (subforests) and apply crossover and mutation operations over 20 iterations. The number of trees for these subforests are randomly determined in the range of 1 to 100. Thus the expected number of trees in each of these subforests will be: $\frac{(1+100)}{2} = 50.5 \approx 50$. Accordingly, the total number of trees involved in each iterations will be: $20 \times 50 = 1000$. All of these 1000 trees are involved in voting on the training data set with 1000 records in 2 separate elitist operation (see Steps 2 and 3 of Algorithm 1). So, the total number of voting events in each iteration can be approximated as: $1000 \times 1000 \times 2 = 2,000,000$. Finally, the total number of added voting events from 20 iterations of Sub_{PGA} can be approximated as: $2,000,000 \times 20 = 40,000,000$. On the other hand, prediction from Sub_{PGA} can save approximately 50 voting for a single test record. Consequently, to overcome 40,000,000 voting Sub_{PGA} needs to predict $\frac{40,000,000}{50} = 800,000$ test records. In today's context, this number is not overwhelmingly large. For example, according to AccuWeather Inc. [19] (the global leader in weather information and digital media), it receives and processes more than 9.5 billion digital requests for it's accurate weather forecasts every day from locations all over the world. This phenomenon verifies the fact that the need for more accurate and efficient classifier is becoming very crucial with such growth in user participation to similar services. Furthermore, the importance of efficient classifiers is reflected in some recent studies [21,49,52].

6. Conclusion

In this paper we propose a technique for selecting a small subforest from a large forest. Often many trees in a forest are not very useful in increasing the ensemble accuracy of the forest. Moreover, if the number of trees in a forest is huge then the computational overhead of predicting future records can be very high, particularly when the number of future records is big. Therefore, it is important to find an effective subforest in order to reduce computational overhead and also sometimes increase ensemble accuracy. A combination of relatively diverse and accurate trees can form a subforest with high accuracy and low size, but we need the right number of trees in the subforest to make it optimal. The number of trees in an optimal subforest can vary significantly from data set to data set and can be difficult for a user/data miner to guess in advance. As a result, we propose a GA-based technique to automatically identify the number of trees to obtain optimal/near optimal subforest.

Table 8 shows us that despite having big numbers of trees compared to the proposed Sub_{PGA}, $Sub_{A-\alpha.D+\delta}$ and $Sub_{A-2\alpha.D+2\delta}$ do not achieve as high accuracy as Sub_{PGA}. Hence, just adding a big number of trees may not help us to get a good ensemble accuracy, we need to add the right combination of right number of trees. The big drop of the number of trees in Sub_{PGA} without dropping the ensemble accuracy is a huge strength of the proposed Sub_{PGA} that does not require any user input to determine the right number of trees of a subforest with high accuracy. Also, one may observe from Table 8 that Sub^{40}_{IA}, Sub^{40}_{ID} and Sub^{40}_{IC} produce similar number of trees as Sub_{PGA} with comparable accuracy, but they need to realize that the numbers of trees in these subforests are user defined and not found automatically. It is interesting to observe that these subforest selection techniques do not achieve as high accuracy as Sub_{PGA} even when they use 80 trees compared to below 45 trees used by Sub_{PGA}. Clearly, Sub_{PGA} shrinks the forest heavily maintaining the best prediction accuracy (even better than the whole forest) for both Bagging and Random Forest (see Table 8) ensembles.

The proposed Sub_{PGA} does not only outperform the greedy approaches it also outperforms (i.e. achieves higher accuracy and lower number of trees) an existing GA-based subforest selection technique called Sub_{HGA}. Fig. 5 suggests that the proposed Sub_{PGA} achieves statistically significant superiority (in terms of ensemble accuracy) over all other techniques including Sub_{HGA}. Obviously, the main difference between Sub_{HGA} and Sub_{PGA} is the novel initial population selection approach of Sub_{PGA}. Hence, our experiments support the usefulness of the proposed initial population selection approach, which is also logically expected (see Section 4.1).

In a nutshell, the proposed Sub_{PGA} successfully uses our initial population selection approach for a GA-based subforest selection and achieves higher accuracy with notably lower number of trees than the complete forest. It also outperforms some existing techniques (including a GA-based technique) and the margins are statistically significant. Hence, the proposed technique achieves our primary goals. However, our future research plan includes further tuning of genetic operations such as Crossover, Mutation and Elitist operation of Sub_{PGA}, and experimental evaluation on more data sets and GA-based approaches.

References

[1] J. Abellan, Ensembles of decision trees based on imprecise probabilities and uncertainty measures, Inf. Fusion 14 (2013) 423–430.
[2] M.F. Amasyali, O.K. Ersoy, Classifier ensembles with the extended space forest, IEEE Trans. Knowl. Data Eng. 16 (2014) 145–153.
[3] S. Arlot, A survey of cross-validation procedures for model selection, Stat. Surv. 4 (2010) 40–79.
[4] A.H. Beg, M.Z. Islam, Genetic algorithm with novel crossover, selection and health check for clustering, in: The 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2016, pp. 575–580.
[5] S. Bernard, S. Adam, L. Heutte, Dynamic random forests, Pattern Recognit. Lett. 33 (2012) 1580–1586.
[6] S. Bernard, L. Heutte, S. Adam, Forest-RK: a new random forest induction method, in: Advanced Intelligent Computing Theories and Applications, in: Lecture Notes in Computer Science, 5227, 2008, pp. 430–437.
[7] V. Bhatnagar, M. Bhardwaj, S. Sharma, S. Haroon, Accuracy-diversity based pruning of classifier ensembles, Progr. Artif. Intell. 2 (2014) 97–111.
[8] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[9] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[10] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth International Group, CA, U.S.A., 1985.
[11] J. Daniel, Sampling Essentials: Practical Guidelines for Making Sampling Choices, SAGE Publications, 2012.
[12] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[13] T. Dietterich, Ensemble methods in machine learning Published, in: Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2000, pp. 1–15.
[14] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (2006) 3–42.
[15] D. Gopika, B. Azhagusundari, A novel approach on ensemble classifiers with fast rotation forest algorithm, Int. J. Innovative Res. Comput. Commun. Eng. 2 (2014) 5380–5387.
[16] J. Han, M. Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, 2006.
[17] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 832–844.
[18] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press (1992).
[19] AccuWeather Inc., Accuweather global weather center, 2015 http://www.accuweather.com/en/press/43009943). Last Accessed: 30/11/2015.
[20] Y.W. Kim, I.S. Oh, Classifier ensemble selection using hybrid genetic algorithms, Pattern Recognit. Lett. 29 (2008) 796–802.
[21] J. Kozak, U. Boryczka, Multiple boosting in the ant colony decision forest meta–classifier, Knowl. Based Syst. 75 (2015) 141–151.
[22] L.I. Kuncheva, Using diversity measures for generating error-correcting output codes in classifier ensembles, Pattern Recognit. Lett. 26 (2005) 83–90.
[23] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with ensemble accuracy, Mach. Learn. 51 (2003) 181–207.
[24] L.A. Kurgan, K.J. Cios, Caim discretization algorithm, IEEE Trans. Knowl. Data Eng. 16 (2004) 145–153.
[25] J. Li, H. Liu, Ensembles of cascading trees, in: Proceedings of the Third IEEE International Conference on Data Mining, 2003, pp. 585–588.

[26] M. Lichman, UCI machine learning repository, 2013 http://archive.ics.uci.edu/ml/datasets.html). Last Accessed: 01/10/2015.
[27] Y. Liu, X.W.X.W. Shen, Automatic clustering using genetic algorithms, Appl. Math. Comput. 218 (2011) 1267–1279.
[28] D.H. Lobato, G.M. Munoz, A. Suarez, How large should ensembles of classifiers be? Pattern Recognit. 46 (2013) 1323–1336.
[29] Z. Lu, X. Wu, X. Zhu, J. Bongard, Ensemble pruning via individual contribution ordering, in: Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2010, pp. 871–880.
[30] D.D. Margineantu, T.G. Dietterich, Pruning adaptive boosting, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 211–218.
[31] R. Mason, D. Lind, W. Marchal, Statistics: An Introduction, Brooks/Cole Publishing Company, 1998.
[32] G.M. Munoz, D.H. Lobato, A. Suarez, An analysis of ensemble pruning techniques based on ordered aggregation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 245–259.
[33] G.M. Munoz, A. Suarez, Aggregation ordering in bagging, in: Proceedings of IASTED International Conference on Artificial Intelligence and Applications, 2003, pp. 258–263.
[34] G.M. Munoz, A. Suarez, Pruning in ordered bagging ensembles, in: Proceedings of the 23rd International Conference on Machine learning, 2006a, pp. 609–616.
[35] G.M. Munoz, A. Suarez, Using boosting to prune bagging ensembles, Pattern Recognit. Lett. 28 (2006b) 156–165.
[36] I. Partalas, G. Tsoumakas, I. Vlahavas, Focused ensemble selection: a diversity-based method for greedy ensemble selection, in: Proceedings of the 18th European Conference on Artificial Intelligence, 2008, pp. 117–121.
[37] R. Polikar, Ensemble based systems in decision making, IEEE Circ. Syst. Mag. 6 (2006) 21–45.
[38] M.A. Rahman, M.Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with k-means, Knowl. Based Syst. 71 (2014) 345–365.
[39] J.J. Rodriguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: a new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1619–1630.
[40] D. Ruta, B. Gabrys, Classifier selection for majority voting, Inf. Fusion 6 (2005) 63–81.
[41] C.A. Shipp, L.I. Kuncheva, Relationships between combination methods and measures of diversity in combining classifiers, Inf. Fusion 3 (2002) 135–148.
[42] C. Tamon, J. Xiang, On the boosting pruning problem, in: Proceedings of the 11th Conference on Machine Learning, 1997, pp. 404–412.
[43] P.N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Pearson Education, 2006.
[44] E.K. Tang, P.N. Suganthan, X. Yao, An analysis of diversity measures, Mach. Learn. 65 (2006) 247–271.
[45] M.F. Triola, Elementary Statistics, Addison Wesley Longman Inc., 2001.
[46] D. Whitley, A genetic algorithm tutorial, Stat. Comput. 4 (1994) 65–85.
[47] F. Wilcoxon, Individual comparison by ranking methods, Biometrics 1 (1945) 80–83.
[48] X. Zeng, D.F. Wong, L.S. Chao, Constructing better classifier ensemble based on weighted accuracy and diversity measure, Sci. World J. 2014 (2014) 1–12.
[49] H. Zhang, F. Min, Three-way recommender systems based on random forests, Knowl. Based Syst. 91 (2016) 275–286.
[50] L. Zhang, P.N. Suganthan, Random forests with ensemble of feature spaces, Pattern Recognit. 47 (2014) 3429–3437.
[51] Y. Zhang, S. Burer, W.N. Street, Ensemble pruning via semi-definite programming, J. Mach. Learn. Res. 7 (2006) 1315–1338.
[52] Q. Zhou, H. Zhou, T. Li, Cost-sensitive feature selection using random forest: Selecting low-cost subsets of informative features, Knowl. Based Syst. 95 (2016) 1–11.
[53] Z.H. Zhou, W. Tang, Selective ensemble of decision trees, in: Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Springer, 2003, pp. 476–483.