Pattern Generation through Feature Values Modification and Decision Tree Ensemble Construction

M. A. H. Akhand, *Member IACSIT*, M. M. Hafizur Rahman, and K. Murase

***Abstract*—An ensemble method produces diverse classifiers and combines their decisions for ensemble's decision. A number of methods have been investigated for constructing ensemble in which some of them train classifiers with the generated patterns. This study investigates a new technique of training pattern generation that is easy and effective for ensemble construction. The method modifies feature values of some patterns with the values of other patterns to generate different patterns for different classifiers. The ensemble of decision trees based on the proposed technique was evaluated using a suite of 30 benchmark classification problems, and was found to achieve performance better than or competitive with related conventional methods. Furthermore, two different hybrid ensemble methods have been investigated incorporating the proposed technique of pattern generation with two popular ensemble methods bagging and random subspace method (RSM). It is found that the performance of bagging and RSM algorithms can be improved by incorporating feature values modification with their training processes. Experimental investigation of different types of modification techniques finds that feature values modification with pattern values in the same class is better for generalization.**

***Index Terms*—Decision tree ensemble, diversity, feature values modification, generalization, pattern generation.**

I. Introduction

The goal of ensemble construction with several classifiers is to achieve better generalization ability over individual classifiers. The inspiration for building an ensemble is the same as for establishing a committee of people: each member of the committee should be as competent as possible, but the members should be complementary to one another. If the members are not complementary, i.e., if they always agree, the committee is unnecessary as any one member could perform the task of the committee. If the members are complementary, then when one or a few members make an error, there is a high probability that the remaining members can correct his error. Thus, for ensemble construction, proper diversity among classifiers (also called base classifiers) is considered to be an important parameter so that the failure of one may be compensated by others [1], [2].

An ensemble method produces diverse classifiers and combines their decisions for ensemble's decision. As a base classifier, decision trees (DTs) are one of the most commonly used methods because they are efficient [3], [4]. Considerable work has been done to determine the effective ways for constructing diverse DTs so that the benefit of ensemble construction could be achieved. There are many ways, such as using different training sets and learning methods, one can adopt to construct diverse DTs. It is argued that DTs construction using different data is likely to maintain more diversity than other approaches [4]-[6] because function that a DT determines approximates from the training data. A number of methods have also been investigated to create different data sets for proper diversity. Among them some manipulate original training data only [5]-[9]; other methods generate some training patterns and a particular classifier is trained with the generated patterns along with the original patterns [10]-[12]. Recently, a number of hybrid ensemble methods have also been investigated incorporating and updating popular methods [13]-[17].

Bagging [5] and boosting [6], the pioneer and popular ensemble methods, sample training patterns from the original patterns to create training sets for different classifiers [6]-[8]. A particular pattern in a particular training set is a copy of a pattern from the original training set that has been selected probabilistically. Bagging creates a training set from an original training set using the bootstrap sampling technique. For a training set, patterns are randomly picked from the original training set with replacement. Due to random selection, each created training set contains many patterns appearing multiple times while others are left out [5]. The boosting algorithm also follows the bootstrap technique to create a training set for a DT. However, the distribution of patterns changes after training each DT. Training patterns that were predicted incorrectly by previous component DT(s) are chosen more often than patterns that were correctly predicted.

The prominent methods those use generated or modified patterns to promote diversity are Random Subspace Method (RSM)[11], Class Label Switching (CLS)[12], and Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples (DECORATE)[10]. RSM constructs a classifier with sampled feature subset of original features of the problem. It uses the bootstrap sampling technique to select features for a classifier and the number of features is considered as half of original in general. Due to bootstrap sampling, a particular training set contains a different arrangement of features; and therefore RSM may produce poor DTs when some important features are missing in their training sets.

To prepare a training set for a particular DT, CLS [12] randomly changes the original class definition of some patterns to different ones. Class label alteration gives new generated patterns in the training set and the generated patterns may conflict with other patterns due to random

Manuscript received December 23, 2012; revised June 1, 2013.

M. A. H. Akhand is with the Dept. of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna-9203, Bangladesh (e-mail: akhand@cse.kuet.ac.bd).

M. M. Hafizur Rahman is with Dept. of Computer Science, KICT, International Islamic University Malaysia, Jalan Gombak, 50728 Selayang, Selangor, Malaysia (e-mail: hafizur@iium.edu.my).

K. Murase is with Graduate School of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan (e-mail: murase@u-fukui.ac.jp).

alteration. In short, CLS may define different class definitions into different training sets for a particular feature set. It may also show the same class label for training patterns with dissimilar feature sets in a particular training set. Both these problems may give rise to ambiguity in training and return poor DTs. Therefore, it has been suggested that the class label should change for a small fraction of examples, leaving a majority of examples with the correct class label [18].

DECORATE[10] randomly creates a set of patterns, called diversity set, for each classifier and it trains a classifier with the union of the original training set and the diversity set. It follows a two-step process to create a pattern: 1) it generates feature values randomly (for continuous feature) and probabilistically (for discrete feature), and 2) it defines the class label of the generated feature set opposite of the class probability response of the existing ensemble. The generated patterns may conflict with original patterns, and therefore, may return poor classifier. To overcome this, it produces a relatively large number of classifiers and selects a subset of classifiers for the final ensemble. The method seems complicated and its computational cost is relatively high due to several steps for pattern generation, relatively large number of classifiers creation and selection of classifiers [18].

With the above discussed DT based ensemble methods, a number of ensemble methods have also been investigated with neural network (NN) as base classifier [7], [18]. Some NN based ensemble methods share similar techniques of DT ensembles and some other are only for NN, such as, negative correlation learning [18], [19]. Recently an ensemble of NN, called EIVA, has been investigated through alteration of preprocessed numeric input values of NN [20]. EIVA does not create training patterns like DECORATE but it manipulates preprocessed numeric input values of some training patterns. The method is shown competitive to the other methods.

The objective of this study is to investigate a best suited pattern generation mechanism for DT ensemble construction that emphasizes both accuracy and diversity among individual DT classifiers. The technique which follows this study is simple and effective: it creates a pattern from an original available pattern modifying some of its feature values with that of another pattern. The DT ensemble construction through the proposed pattern generation seems an easy and effective method. Moreover, this study also investigates hybrid ensemble methods incorporating such feature values modification technique with other existing methods. The experimental results on a large number of benchmark problems reveal the effectiveness of generated patterns using the proposed technique in DT ensemble construction.

The rest of this paper is organized as follows. Section II describes the proposed feature values modification for pattern generation and explains ensemble of decision tree construction with the generated patterns. Section III explores hybrid ensemble methods incorporating proposed feature values modification with two popular methods. Section IV is for experimental study. Finally, the paper concludes with a few remarks in Section V.

II. Pattern Generation through Feature Values Modification and Decision Tree Ensemble Construction

This section explains proposed pattern generation modifying feature values and a Decision Tree Ensemble (DTE) method based on this technique. The basic aim of pattern generation is the creation of different training sets for different classifiers to promote diversity in an ensemble. Pattern generation seems inefficient in the existing methods to maintain proper diversity for better generalization. Some of them (e.g., DECORATE) follow several steps to create patterns and are computationally heavy. Some of them (e.g., CLS) produce very different patterns and, therefore, may return poor classifier due to ambiguity in training. It is reported that one may get very good diversity producing very different patterns from any of the existing methods, but achieving a good generalization is not easy [2], [18]. For better generalization, classifiers should be accurate as well as diverse. Accordingly, this study investigates a pattern generation mechanism that uses existing patterns to produce different but related patterns, and may maintain the accuracy of individual classifier to deliver proper diversity. Fig. 1 shows the algorithm for the proposed Feature Values Modification (FVM) that introduces some generated patterns in a particular training set and explains the important points here after.

Function *FVM* (T, T_d, *MF*)
{ // T and T_d are the original training set and the training set for a particular DT, respectively.
//*MF* is the Modification Factor.

1. Number of patterns to be modified $M=S*MF$ // Here S is the size of T or T_d
Prepare Pattern Index List P_{Index} with values 1 to S.

$S_{Index}= S$ // Size of Pattern Index List is equal to S

2. For m = 1 *to* M {

a. Select a random number I between 1 and S_{Index}

b. Select Pattern Index m from I_{th} location of P_{Index}

c. Delete I_{th} location of P_{Index} // $S_{Index}= S_{Index}$-1

d. Select m_{th} pattern P_m in T_d that would be modified

e. Select a pattern P_o ($P_o \neq P_m$) in T having same class label of P_m

f. Modify some feature values of P_m using corresponding values from P_o }

}

Fig. 1. Feature Values Modification (FVM) in a Given Training Set.

FVM of Fig. 1 takes original training set (T) and training set for a particular DT (T_d), and modifies feature values of some patterns in T_d with help of patterns of T to induce generated patterns. The number of patterns to be modified/ generated depends on the parameter Modification Factor (*MF*) that values may define between 0.0 and 1.0. To select a pattern for modification only once, a pattern index list (P_{Index}) is used and the index of a pattern is deleted after selection (Step 2. c). A pattern would be modified (P_m) is selected randomly through generating a random number of pattern index I in between 1 and current size of P_{Index} (Step 2. a and Step 2. b). To produce a new pattern FVM alters some feature

values of a pattern P_m of T_d with corresponding values of another selected pattern P_o from the original training set T. The pattern P_o is considered from T to increase versatility of selection in feature and feature values. In general, P_o will have same class label of P_m so that generated patterns will be different but related to original patterns. However it will be possible to generate a pattern modifying its feature values with a pattern having a different class label. Next important points are how many feature values will be changed among the total features (f) and what will be the procedure for it. A crossover like technique is employed in this study for this purpose: randomly select a point (cp) in between 1 and f-1; and then feature values from cp+1 to f of P_m is changed with that of P_o.

The feature values modification technique may produce a large number of patterns from the existing patterns. Consider a particular problem to classify N patterns into c different classes based on f input feature values and each class have N_c (i.e., $N_c = N/c$) patterns. In such a case, a pattern may consider other (N_c-1) patterns to modify its feature values and may produce (f-1) different patterns for each of individual pattern selecting cp value in between 1 and f-1. Therefore, the possible total number of generated patterns (Gp_{sc}) modifying a pattern with another same class label pattern is:

$$Gp_{sc} = cN_c(N_c\text{-}1)(f\text{-}1) = N(N_c\text{-}1)(f\text{-}1). \quad (1)$$

Similarly, a pattern may consider (N-N_c) different patterns for modification with different class label patterns and in such a case all f feature values modification also give new patterns. The possible number of generated patterns (Gp_{dc}) for modification with different class label pattern is:

$$Gp_{dc} = cN_c(cN_c\text{-}N_c)f = cN_cN_c(c\text{-}1)f = N_cN(c\text{-}1)f \quad (2)$$

On the other hand, CLS may produce c-1 different patterns from an original pattern changing its class label to one of remaining c-1 classes. For a two class problem, CLS may produce only one pattern from an original one. The possible total number of generated patterns in CLS may be:

$$Gp_{CLS}=cN_c(c\text{-}1)=N(c\text{-}1). \quad (3)$$

Thus, FVM technique may produce more patterns than CLS. Moreover, FVM technique may generate patterns like CLS modifying all f feature values with a different class label pattern.

The way of pattern generation modifying feature values in FVM is easy and seems to be cost effective. FVM changes some feature values of a pattern with that of another pattern. This easy technique is applicable for any feature type and no need to check whether the feature is numeric or discrete. On the other hand, DECORATE follows different ways to generate feature values for different feature types and incurs a separate step to define class label for the generated feature set.

Another benefit of FVM is its simplicity. The parameters used in it are easily understandable and the selection of their values is also simple. The parameter MF determines the number of patterns to be generated. The higher value of MF may give more diversity than its lower value due to the larger number of different generated patterns in the training sets. Moreover, the technique of pattern generation of FVM can be hybridized with other training schemes like the one used in the bagging and RSM. This issue will be explored in Section III.

1. Take the original training set $T = \{P_1 \ldots, P_N\}$ each pattern belongs feature values from feature set $F = \{1, 2, \ldots, f\}$ and class label $c_p \in C = \{1, 2, \ldots, c\}$ // Problem to classify N patterns into c different classes based on f input feature values.
Let D be the number of DTs to be constructed and
MF be the Modification Factor
2. *For d = 1 to D* {
a. $T_d = T$ // Take original training set
b. ***FVM (T, T_d, MF)* // Call Function FVM**
c. Construct decision tree DT_d with T_d}
3. Ensemble decision from D decision trees.

Fig. 2. Ensemble based on Feature Values Modification (EFVM).

Fig. 2 shows the algorithm for the proposed ensemble of decision trees conceiving the pattern generation of FVM described in Fig.1 and the new ensemble method is called as Ensemble with Feature Values Modification (EFVM). To create a different training set for a particular DT, EFVM first takes the original training set (Step 2. a) and then call FVM (Step 2. b) to alter some of its patterns with newly generated patterns. EFVM constructs several DTs with different training set each of which contains some generated patterns through FVM and considers all the produced DTs for an ensemble. EFVM does not require DT selection like DECORATE that produces a relatively large number of DTs and consider a subset of them for final ensemble [10].

The number of patterns to be generated in the training set of a particular DT in EFVM is specified by the Modification Factor (MF). The value of MF may set anywhere between 0.0 and 1.0. No pattern will be changed and all the DTs will be trained on T for MF = 0.0; for MF = 1.0 all the patterns are updated, and there might be no pattern common in between the original training set (T) and the training set of a particular DT (T_d).

III. Hybrid Ensemble Construction incorporating Feature Values Modification

This section explains hybrid ensemble construction methods incorporating the technique of pattern generation of FVM with other data sampling techniques. From various existing methods, bagging and random subspace method (RSM) have been considered due to their popularity as well as effectiveness. Bagging creates a separate training set for each DT in the ensemble using resampling technique from the original training data. On the other hand, RSM also produces different training sets for different DTs but a particular training set contains a sampled subset of original feature set. The aim of separate training set creation using sampling data in bagging and feature in RSM is to produce diverse DTs so that ensemble with them achieves better generalization. Due to random selection from the original training set, each created training set in bagging contains on average 63.2% unique patterns from the original training set,

with many patterns appearing multiple times while others are left out [5]. This means diversity produces by bagging might not be so precise, and it could be made more precise introducing FVM in bagging. Similarly, the incorporation of FVM may improve RSM since training sets of it may contain common feature subset.

1. Take the original training set $T= \{P_1...,P_N\}$ each pattern belongs feature values from feature set $F = \{1, 2,..., f\}$ and class label $c_p \in C = \{1, 2,...,c\}$ // Problem to classify N patterns into c different classes based on f input feature values.
Let D be the number of DTs to be constructed and
MF be the Modification Factor
2. *For d = 1 to D* {
a. Make a new training set, T_d by sampling N patterns uniformly at random with replacement from T.
b. ***FVM* (*T*, T_d, *MF*) // Call Function FVM**
c. Construct decision tree DT_d with T_d}
3. Ensemble decision from D decision trees.

Fig. 3. Hybrid Ensemble: Bagging with Feature Values Modification (BFVM).

Fig. 3 shows the pseudo code of hybrid ensemble method incorporating FVM with bagging and the new method is called Bagging with Feature Values Modification (BFVM). The bold-face line in the figure (Step 2.b) indicates the modifications/additions over standard bagging algorithm. The hybrid BFVM method first prepare sampled training set for particular DT (T_d) as of bagging (Step 2.a) and then call FVM (Step 2.b) to alter some of its patterns with newly generated patterns. Step 2.b is only addition to the standard bagging method to hybridize the feature values modification. The difference between BFMV and EFVM (Fig. 2) is that BFVM considers bagging like sampled training set for feature values modification whereas EFVM takes a copy of original training set (T_i=T).

1. Take the original training set $T= \{P_1...,P_N\}$ each pattern belongs feature values from feature set $F = \{1, 2,..., f\}$ and class label $c_p \in C= \{1, 2,...,c\}$ // Problem to classify N patterns into c different classes based on f input feature values.
Let D be the number of DTs to be constructed and
MF be the Modification Factor
2. *For d=1 to D* {
a. $T_d = T$ // Take original training set
b. ***FVM* (*T*, T_d, *MF*) // Call Function FVM**
c. Make a sampled feature subset F_d from original feature set F.
d. Update T_d removing features that is not in F_d
e. Construct decision tree DT_d with T_d}
3. Ensemble decision from D decision trees.

Fig. 4. Hybrid Ensemble: Random Subspace with Feature Values Modification (RFVM).

Similarly, Fig. 4 is the pseudo code of hybrid ensemble method of RSM with Feature Values Modification (RFVM) incorporating FVM with RSM. The bold-face line in the figure (Step 2. b) indicates the modifications/additions over standard RSM algorithm. Since RSM requires different feature subsets for different DTs, the feature values modification (Step 2. b) is done on the copy of original training set (T_d=T) (Step 2. a) before feature sampling for simplicity. A particular training set for a particular DT is prepared considering sampled feature subset of T_d that already contains some generated patterns. In RFVM, Step 2.b is the only addition over standard RSM to hybridize FVM with RSM. The difference between EFVM (Fig. 2) and the hybrid RFVM is that RFVM uses a sampled feature subset whereas EFVM uses all the features for a particular DT.

IV. Experimental Studies

This section experimentally investigates the proficiency of proposed feature values modification based pattern generation for DT ensemble construction. A set of benchmark problems were chosen as a test bed and the performance were compared to that of other ensemble methods: bagging, Random Subspace Method (RSM), Class Label Switching (CLS) and DECORATE. For fair comparison, the experimental methodology was chosen carefully. Finally, an experimental analysis is given to observe the variation effect of *MF* value on diversity and generalization of an ensemble.

A. Benchmark Problems and General Experimental Methodology

TABLE I: Characteristics of Benchmark Datasets

Sl.	Dataset	Exam-ple	Class	Input Feature	
				Cont.	Disc.
1	Australian Credit Card	690	2	6	9
2	Auto (ATO)	205	6	16	10
3	Breast Cancer Wisconsin (BCW)	699	2	9	-
4	BUPA Liver Disorders (BLD)	345	2	6	-
5	Diabetes (DBT)	768	2	8	-
6	Ecoli (ECL)	336	4	7	-
7	German Credit Card (GCC)	1000	2	7	13
8	Glass (GLS)	214	6	9	-
9	Heart Disease Cleveland (HDC)	303	2	6	7
10	Hepatitis (HPT)	155	2	6	13
11	Hypothyroid (HTR)	7200	3	6	15
12	Ionosphere (INS)	351	2	34	-
13	Iris Plants (IRP)	150	3	4	-
14	King+Rook vs King+Pawn (KRP)	3196	2	-	34
15	Lymphography (LMP)	148	4	-	18
16	Lung Cancer (LNG)	32	3	-	56
17	Lenses (LNS)	24	3	-	4
18	Letter (LTR)	20000	26	16	-
19	Optical Rec.of Handwritten Digits (OPT)	5620	10	64	-
20	Page Blocks (PGB)	5473	5	10	-
21	Phoneme (PHN)	5404	2	5	-
22	Postoperative (PST)	90	3	1	7
23	Soybean (SBN)	683	19	-	35
24	Sonar (SNR)	208	2	60	-
25	Splice (SPL)	3175	3	-	60
26	Vehicle (VHC)	846	4	18	-
27	Wine (WIN)	178	3	13	-
28	Waveform (WVF)	5000	3	21	-
29	Yeast (YST)	1484	10	8	-
30	Zoo (ZOO)	101	7	15	1

Thirty real world classification problems were employed in this study. The origin of these problems is the machine learning benchmark repository at the University of California, Irvine (UCI); detailed descriptions are available at the UCI website [21]. Table I shows the characteristics of the problems which show a considerable variety in the number of examples, input features, and classes. Thus, these problems provide a suitable experimental test bed.

In order to evaluate the performance of an ensemble, generalization was measured on a testing set that was reserved from available data and not used by any DT in an ensemble. The testing error rate (TER) is the common measure of generalization: the lower its value, the better is the generalization. Note that the aim of any ensemble method is to minimize the TER. It can be seen that the TER may vary due to the variation of the testing data, even if the size of the data set remains the same. Therefore, standard 10-fold cross validations have been used for result presentation [7]-[12]. In the cross validation, initially available training patterns were partitioned into 10 equal or nearly equal sets, and for each turn, one set was reserved as a testing set, while the remaining nine sets were used for constructing DTs. However, different sizes of training and testing set partitions may give different results.

We followed a common general experimental setup that does not favor any particular method. Variation of built-in parameter values to achieve better result from an ensemble method is very common. DECORATE was tested varying R_{Size} values from 0.25 to 0.75; CLS was tested with $S_{Fraction}$ values from 0.1 to 0.3; and EFVM considered *MF* values from 0.4 to 0.8. Although RSM has found effective with one half of total features [11], we also tested with 75% features to increase chance to get better result with RSM. The best result for a method varying the above parameters was used to compare with the other methods. The algorithms are implemented on Visual C++ of Visual Studio 2010 with Weka, the popular free machine learning tool. The experiments have been done on a PC (Intel Core i3-2100 @3.10 GHz CPU, 2GB RAM) with Windows 7 OS. A fixed number of 10 DTs were constructed for an ensemble, except for DECORATE. Previous studies have revealed that ensemble with this number of classifiers is able to show good generalization [7], [18], [19]. To be comparable to the other methods, the maximum number of DTs per ensemble in DECORATE was defined as 10 and the maximum number of trial DTs was 15.

B. Managing Weka for Base Classifier Construction

This study investigates efficiency of pattern generation through feature values modification for ensemble construction and therefore the method of base classifier (i.e., DT) construction was common for all the ensemble methods for proper understanding. We used in this study Weka (Waikato Environment for Knowledge Analysis), the popular suite of machine learning free software developed at the University of Waikato, New Zealand [22], [23]. From various models in Weka we employed j48 model for DT construction based on C4.5 [24] algorithm. C4.5 is the well known and popular DT building algorithm and many of previous studies employed it [11], [12]. Instead of implementing C4.5 or any other algorithm by ourselves, the use of a third party standard method for constructing DT seems more appreciative to justify the efficiency of an ensemble building technique. Such technique also helps in easy implementation of ensemble methods.

We managed Weka3.7.3, a latest version of Waka, for base DT construction. We setup Weka3.7.3 and copied the *weka.jar* file to our working application folder for easier operation. A number of DTs are constructed using Weka for an ensemble but we provided different training sets for different DTs. Since Weka only considers *arrf* file type, we produced and saved training set (*TrnData.arrf*) and testing set (*TstData.arrf*) in specified *arrf* file format in a defined Path location. Then *Statement 1* is executed to build a DT model on the *TrnData.arrf* and store in the model for later use in the defined location. In the statement *–t* specifies the training file; and –no-cv-d is for no cross validation and model will be saved as the specified location and name. It is notable that the cross validation matter has been managed from the upper level and outside Weka. Finally, *Statement 2* is executed to test the DT model on the *TstData.arrf* and store the output as *TstOutput.txt* file in the specified location. In the statement, *–T* specifies the test file; -l is to load previously saved model; and *–p* 0 > defines output file format. For ensemble construction with 10 DTs, 10 DT models will be produced from 10 different training sets (i.e., *TrnData.arff* files) and ensemble performance (i.e., generalization) will be measured incorporating 10 different *TstOutput.txt* files.

Statement 1: To create a model from the training data

java -classpath weka. jar weka. classifiers. trees. J48 *-t Path/TrnData.arff* -no-cv-d *Path/ModelName.model*

Statement 2: To evaluate the created model on testing data

java -classpath weka. jar weka. classifiers. trees. J48 *-T Path/TstData.arff* -l *Path/ ModelName.model* -p 0 > *Path/TstOutput.txt*

C. Performance of EFVM and Comparison with Conventional Methods

This section evaluates the performance of EFVM, the ensemble method based on feature values modification (FVM), compares with bagging, RSM, CLS and DECORATE ensemble methods. The experimental results of hybrid ensembles incorporating the feature values modification are also presented in this section. For generalization comparison, results presented are the average TERs over five standard 10-fold cross-validation (i.e., $5 \times 10 = 50$) runs.

Table II presents TERs achieved by bagging, RSM, CLS, DECORATE and EFVM where the best TER among the five ensemble methods is shown in bold-face type and worst one in italic-underlined for each problem. A pair two tailed *t*-test was conducted between EFVM and other ensemble methods individually to determine the significance in the variation of results for each problem. If TER of an EFVM method is found significantly better than the method by *t*-test, it is marked with a plus (+) sign. On the other hand, a minus (-) sign indicates that the TER of EFVM is significantly worse than the conventional method for a particular problem. A single plus/minus means that the TER difference is statistically significant with 95% confidence interval and a double plus/minus is for 99% confidence interval.

TABLE II: TER COMPARISONS OF EFVM WITH BAGGING, RSM, CLS AND DECORATE OVER FIVE STANDARD 10-FOLD CROSS-VALIDATION RUNS. A PLUS (OR MINUS) SIGN INDICATES THAT TER OF EFVM IS FOUND SIGNIFICANTLY BETTER (OR WORSE) THAN SNNE BY T-TEST. A SINGLE AND DOUBLE PLUS/MINUS IS FOR 95% AND 99% CONFIDENCE INTERVAL, RESPECTIVELY. THE BOTTOM OF THE TABLE CONTAINS A RESULTS SUMMARY OF INDIVIDUAL METHODS AND COMPARISON OF EFVM WITH OTHER METHODS.

Sl.	Problem	TERs achieved by conventional ensemble methods				TER of EFVM	*t*-test of EFVM with conventional methods			
		Bagging	RSM	CLS	DECOR.		Bagg.	RSM	CLS	DEC.
1	ACC	0.142	0.1423	0.1371	*0.1449*	**0.1345**		+		+
2	ATO	*0.234*	0.202	0.197	**0.19**	0.209	+			
3	BCW	0.0409	**0.0403**	0.0409	*0.042*	*0.042*				
4	BLD	0.29	*0.3647*	0.2876	0.3376	**0.28**		++		++
5	DBT	0.2434	0.2416	0.2455	*0.2458*	**0.2311**	+		+	+
6	ECL	0.0945	*0.1006*	0.0891	**0.0867**	0.0885		++		
7	GCC	0.256	0.2598	0.2524	*0.2644*	**0.2514**		+		++
8	GLS	0.3019	0.3257	0.3105	*0.3314*	**0.2886**		+		++
9	HDC	0.2193	*0.222*	0.2027	0.208	**0.19**	++	++		+
10	HPT	0.184	*0.196*	0.168	**0.1613**	0.1747		+		
11	HTR	0.0039	0.0037	*0.004*	0.0033	**0.0032**	++		++	
12	INS	0.0983	**0.072**	*0.0994*	0.084	0.0937		−−		
13	IRP	0.048	0.044	0.0467	**0.04**	*0.0493*				
14	KRP	**0.0201**	*0.0781*	0.0218	0.0212	0.0203		++		
15	LMP	0.1929	*0.2086*	0.2	0.2014	**0.1886**				
16	LNG	0.4867	0.5267	*0.5467*	**0.4733**	0.4933				
17	LNS	0.15	*0.17*	0.15	0.16	**0.11**	+	+	+	+
18	LTR	0.0766	**0.0628**	0.0769	*0.1174*	0.0754		++		++
19	OPT	0.058	**0.0391**	0.067	*0.0967*	0.0546	+	−−	++	++
20	PGB	0.028	*0.0293*	0.0278	0.0284	**0.026**	++	++	++	++
21	PHN	**0.1088**	*0.1409*	0.1386	0.1271	0.1277	−−	++	++	−−
22	PST	*0.3289*	**0.2889**	0.2956	0.3067	0.2933	++			
23	SBN	*0.0797*	0.0765	0.0747	**0.0724**	0.0776				
24	SNR	*0.241*	0.184	0.21	**0.178**	0.206	++			−
25	SPL	0.0567	**0.0536**	0.0551	*0.0593*	0.0558				+
26	VWL	0.3949	*0.4214*	**0.3749**	0.4042	0.3945		+	−	
27	WIN	*0.0835*	0.0682	0.0659	**0.0329**	0.0659	++			−−
28	WVF	0.1818	0.1763	0.1878	*0.1888*	**0.1752**	++		++	++
29	YST	0.398	0.4203	0.4015	*0.437*	**0.3922**		++	+	++
30	ZOO	*0.068*	0.056	**0.048**	0.054	0.056				
Average TER		**0.1703**	**0.1738**	**0.1674**	**0.1699**	**0.1616**				
Best/Worst TER		**2/6**	**6/10**	**2/3**	**8/12**	**12/2**				
Win/Draw/Lose and *t*-test Better/Worse of EFVM		**25/0/5**	**19/1/10**	**21/1/8**	**18/1/11**	**-**	**11/1**	**14/2**	**8/1**	**13/3**

Table II clearly shows the proficiency of EFVM, the DT ensemble with generated pattern through feature values modification technique. EFVM is shown to achieve the lowest average TER of 0.1616; the average TERs for bagging, RSM, CLS and DECORATE are 0.1703, 0.1738, 0.1674 and 0.1699, respectively. Although RSM is shown the worst among the ensemble methods on the basis of average TER, it is shown best for six cases. In general, RSM performed well for the problems having a sufficient number features (e.g., SPL, OPT) since it uses sampled features for constructing DTs. And, when all the features are important for a problem, RSM might perform worse for that problem. This might be the reason to show the worst TERs for several cases such as LNS, VWL and PHN. On the other hand, EFVM uses all the features but promote diversity through generated patterns and performs better than RSM for 19 cases out of 30 cases in which the results are found significant by *t*-test for 14 problems.

From Table II it is also recognized that bagging performs well for problems having large number of patterns (e.g., BCW, PHN) since bootstrap sampling is efficient for larger training set. On the other hand, bagging may perform worse for problems having limited number of patterns (such as WIN, SNR) because in such small sized problem each individual training pattern is important and bootstrap left some original patterns in a training set. On the other hand, EFVM uses all the original training patterns in a training set but a portion of some patterns are altered modifying feature values. Therefore, it is seen from Win/Draw/Lose comparison of Table II that EFVM outperformed bagging for 25 cases out of 30 cases in which the results are found significant by *t*-test for 11 problems.

CLS changes class label that may introduce very different patterns in a training set and therefore perform well for only few problems. DECORATE generates examples that are different from the original patterns and has been found to be good for problems with limited numbers of examples, such as ECL, HPT and SNR. On the other hand, EFVM generates patterns from original patterns modifying feature values and might be difficult to produce patterns for better performance when the problem contains a very limited number of patterns as well as features. Therefore, EFVM has the worst TERs for some small-sized problems with few features such as HPT, IRP and WIN. At a glance, EFVM outperformed CLS and DECORATE for 21 and 18 cases, respectively in which the results are found significant by *t*-test for several cases. EFVM is also shown to achieve best TERs for 12 cases; whereas bagging, RSM, CLS and DECORATE are shown best for 2, 6,

2, and 8 cases, respectively. On the other hand, EFVM is shown the worst TERs for only two cases; whereas bagging, RSM, CLS and DECORATE are shown the worst for 6, 10, 3, and 12 cases, respectively. Overall, EFVM is shown to be a good ensemble construction method, generating different patterns for different DTs through feature values modification.

D. Performance of Hybrid Ensembles Incorporating Feature Values Modification

This section presents experimental results of two hybrid ensemble methods BFVM and RFVM, incorporating feature values modification (FVM) technique with bagging and RSM, described in Section III. Table III presents average TERs over five standard 10-fold cross-validation (i.e., $5 \times 10 = 50$) runs of BFVM and RFVM for 20 problems. The results for standard bagging and RSM have been taken from Table II. Between a standard and a hybrid method, the better TER is marked as bold-face type and a star sign with TER of a hybrid method indicates that TER is better than TER of any individual methods of Table II.

The results presented in Table III clearly indicate the benefit of hybrid ensembles incorporating feature values modification with bagging and RSM. The performance of bagging relies on the diversity due to bootstrap sampling of patterns and it does not have any other parameter to tune. Therefore, when data sampling does not give proper diversity for better generalization for a problem, FVM may help to reach at better generalization position. Out of 20 cases, BFVM outperformed bagging for 19 cases and incorporation of FVM is found ineffective for only for WIN. It is already discussed in the previous section that FVM operation is not suitable for WIN.

TABLE III: TER COMPARISONS AMONG STANDARD BAGGING, STANDARD RSM, WITH BFVM AND RFVM OVER FIVE STANDARD 10-FOLD CROSS-VALIDATION RUNS. A STAR SIGN INDICATES THAT TER OF THE HYBRID METHOD IS BETTER THAN ANY INDIVIDUAL METHODS.

Sl.	Problem	TER achieved by standard and hybrid ensemble methods			
		Bagging	BFVM (Bagg.+FVM)	RSM	RFVM (RSM+FVM)
1	ACC	0.142	**0.1362**	0.1423	**0.1345**
2	BCW	0.0409	**0.0397***	0.0403	**0.0365***
3	BLD	0.29	**0.2882**	0.3647	**0.3135**
4	DBT	0.2434	**0.2355**	**0.2416**	0.245
5	ECL	0.0945	**0.0879**	0.1006	**0.0806***
6	GCC	0.256	**0.246***	0.2598	**0.255**
7	HDC	0.2193	**0.206**	0.222	**0.1993**
8	HPT	0.184	**0.176**	0.196	**0.1467***
9	INS	0.0983	**0.0926**	**0.072**	0.0783
10	IRP	0.048	**0.0453**	**0.044**	0.0533
11	LMP	0.1929	**0.1857***	0.2086	**0.1843***
12	LNG	0.4867	**0.44***	0.5267	**0.4533***
13	LNS	0.15	**0.12**	0.17	**0.12**
14	PGB	0.028	**0.0265**	0.0293	**0.0273**
15	SBN	0.0797	**0.0738**	0.0765	**0.0729**
16	SNR	0.241	**0.218**	**0.184**	0.2
17	SPL	0.0567	**0.0543**	0.0536	**0.0524***
18	WIN	**0.0835**	0.0859	**0.0682**	0.0729
19	WVF	0.1818	**0.1757**	0.1763	**0.1706***
20	ZOO	0.068	**0.058**	0.056	**0.048**
Average TER		**0.1592**	**0.1496**	**0.1616**	**0.1472**

Similarly, RSM only relies on feature sampling and incorporation of FVM may help to tune diversity to give better generalization when feature sampling does not give proper diversity for better generalization for a problem. Out of 20 cases, RFVM outperformed bagging for 15 cases and incorporation of FVM is found ineffective for rest five cases. For the five cases RSM has already shown best or good TER as of Table II (e.g., INS, DBT and SNR) or FVM is not suitable for it (e.g., WIN). Moreover, both BFVM and RFVM are shown better average TERs than corresponding standard methods i.e., bagging and RSM, respectively.

It is also interesting to observe from the Table III that hybrid methods achieved best TERs for some cases when compared with the other methods in Table II. It is seen from Table II that the best TER achieved (i.e., 0.1886) for LMP problem by EFVM. For the same problem bagging and RSM are shown TERs 0.1929 and 0.2086, respectively. Incorporating FVM with bagging and RSM, BFVM and RFVM are shown to achieve TERs 0.1857 and 0.1843, respectively; these values are better than any individual standard method. Comparing results of hybrid methods in Table III with results of Table II it is found that BFVM and RFVM are shown better TERs than the best TERs of Table II for four and seven cases out of tested 20 cases. RSM was the best individual method for SPL problem as of Table II showing TER 0.0536; and FVM incorporation improved its TER to 0.0524. On the other hand, RSM was the worst for HPT problem with TER 0.196 but RFVM is shown to achieve TER 0.1467 that is even better than the best TER (i.e., 0.1613 by EFVM) for individual method. This explanation clearly indentifies the effectiveness of incorporation of FVM with bagging and RSM.

E. Experimental Analysis

This section describes experimental results to explore the effects on diversity and TER of varying parameter values. Three problems were selected for analysis, based on variations in the number of available examples, input features, and output classes. For example, the German Credit Card (GCC) problem contains both continuous and discrete features, whereas the Diabetes (DBT) problem has only continuous features and the Lymphography (LMP) problem has only discrete features.

The diversity indicates how predictions differ among component classifiers (i.e., DTs) on the testing set. To measure diversity, we employed the most commonly used pairwise plain disagreement measure technique [25]. The plain disagreement diversity for a networks pair i and j is given by

$$div_{i,j} = \frac{1}{N}\sum_{n=1}^{N} Diff\,(C_i(n), C_j(n)), \qquad (4)$$

where N is the number of examples in the testing set, $C_i(x_k)$ is the class assigned by DT i to example k, and $Diff(a,b) = 0$ if $a = b$, otherwise $Diff\,(a,b) = 1$. The total ensemble diversity is the average for all DT pairs in the ensemble.

It is already mentioned that the feature values modification (FVM) of a pattern is done with another pattern with same class label and the modification with different class label pattern might give more diversity. The matter explores in this section through experimental results with a modified version of EFVM, called different class EFVM (dcEFVM), where

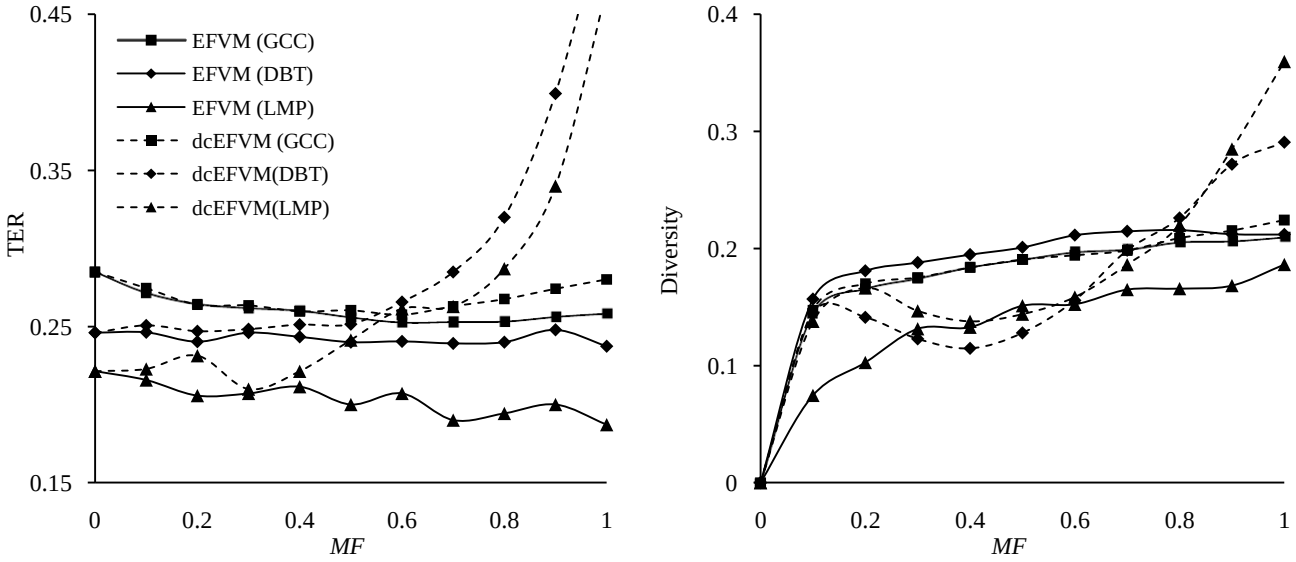

Fig. 5. Effects of the variation of Modification Factor (MF) on TER and Diversity on EFVM and dcEFVM.

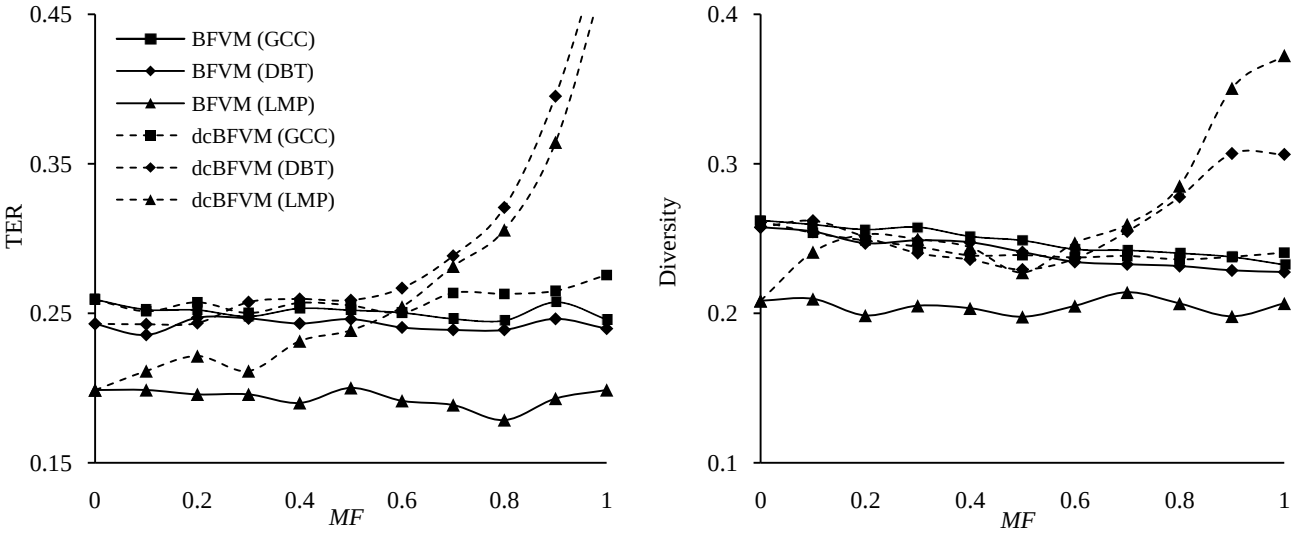

Fig. 6. Effects of the variation of Modification Factor (MF) on TER and Diversity on BFVM and dcBFVM.

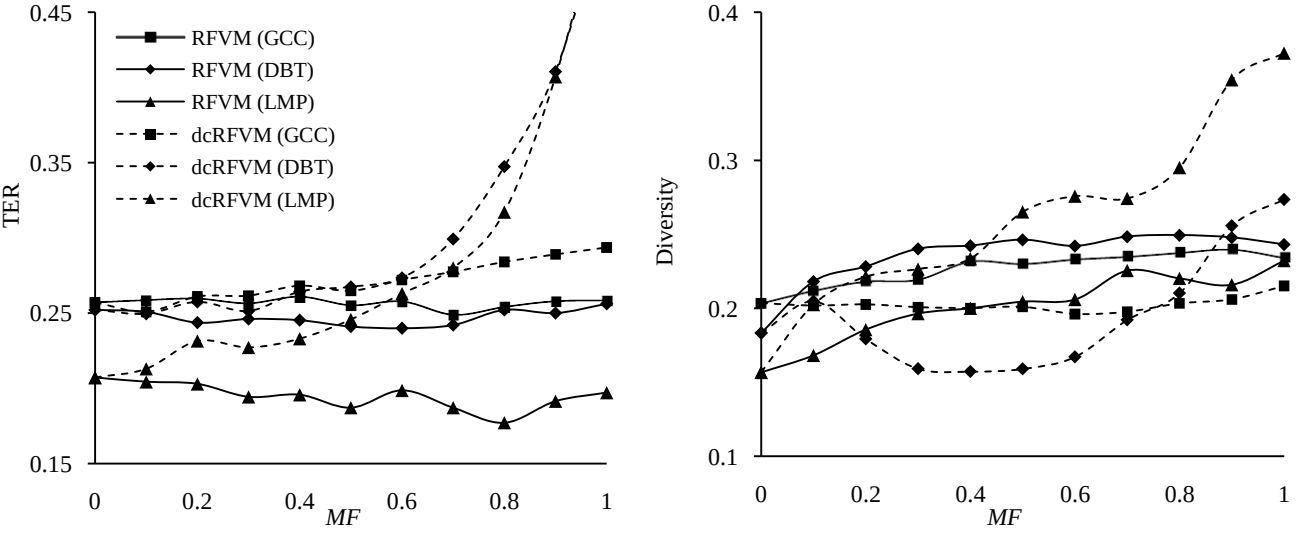

Fig. 7. Effects of the variation of Modification Factor (MF) on TER and Diversity on RFVM and dcRFVM.

modification is done with different class label pattern. Similarly, different class BFVM (dcBFVM) and different class RFVM (dcRFVM) are also considered to observe feature values modification with different class label pattern on hybrid ensemble methods.

Fig. 5 shows TER and diversity achieved by EFVM and dcEFVM over five standard 10-fold cross-validation runs varying modification factor (*MF*). It is already mentioned that the user-defined value of *MF* indicates the number of patterns to be considered to modify the feature values. At

MF=0.0, no pattern were considered to modify and all the DTs are constructed on same original training set. Therefore, for *MF*=0.0, with zero diversity the ensemble act like single DT and achieved worst TER for any problem in general. As an example, TERs at *MF*=0.0 for GCC and LMP problems are 0.285 and 0.2214, respectively; the values are worst among the presented results. On the other hand, diversity increases with *MF* for any problem as can be seen in the figure. And diversity improvement is more visible for dcEFVM than EFVM for any problem. In dcEFVM, the feature values of a pattern are modified with the values from another pattern with a different class label, and this may give a completely different pattern for training.

Completely different pattern is good for diversity, but obtaining decisions from highly diverse DTs is not easy [18]. Therefore, very high diversity is not good for better generalization (i.e., reducing TER) and there is a tradeoff between diversity and TER [26]. This is why performance degraded (i.e., TER increased) rapidly with respect to *MF* for dcEFVM after a certain *MF* value in Fig. 5. On the other hand, FVM with same class pattern is shown to be more effective in achieving better generalization (i.e., lower TER). Feature values modification using patterns from the same class generates similar patterns and does not give so much of the diversity that is not good for TER. From the figure it can be clearly seen that diversity improves smoothly with *MF* for FVM and reduces TER in most of the cases. As an example, for LMP problem TER decreased with diversity improvement when *MF* values are increased from 0.0 and best TER (i.e., 0.1871) achieved at *MF*=1.0. Thus, pattern generation by replacing feature values of patterns with those from other patterns in the same class is a good approach for ensemble construction.

Fig. 6 shows TER and diversity achieved by BFVM and dcBFVM over five standard 10-fold cross-validation runs varying *MF* values. The value of *MF*=0.0 in the figure indicates standard bagging algorithm and the diversity is shown at this point due to data sampling. It is noticeable from Figs. 5(b) and 6(b) that for same *MF* value BFVM and dcBFVM in Fig. 6(b) is shown greater diversity than EFVM and dcEFVM in Fig. 5(b) in most of the cases. This is reasonable because incorporation of FVM may enhance the diversity of bagging. However, rapid diversity improvement in dcBFVM does not shown to improve TER at all. On the other hand, BFVM is shown to achieve better TER than bagging (*MF*=0.0) in general, although some cases diversity slightly reduces in BFVM for larger *MF* value. For GCC problem, the TER was 0.2592 for bagging (at *MF*=0.0) whereas BFVM achieved TER of 0.2466 at *MF*=0.7, the value is also better than any value EFVM of Fig. 5(a). Thus, *MF* parameter of FVM may help to tune proper diversity for better TER in bagging, and BFMV is an effective hybrid ensemble method.

Fig. 7 shows TER and diversity achieved by RFVM and dcRFVM over five standard 10-fold cross-validation runs varying modification factor (*MF*). The value of *MF=0.0* in the figure indicates standard RSM algorithm and the diversity is shown at this point due to feature sampling. Like BFVM, RFVM also shows better TERs than EFVM (Fig. 5(a)) for a value of *MF*. Moreover, with smooth diversity improvement with respect to *MF*, RFVM is shown to achieve better TER than RSM and dcRFVM, in general. Thus, RFVM may give better generalization tuning diversity of RSM with feature values modification and it is shown to achieve best TERs for some cases. As an example, for LMP problem, the TER for RSM (*MF*=0.0) was 0.2071, and RFVM is shown to achieve TER 0.1771 at *MF*=0.8. The achieved TER is also the best considering EFVM and BFVM too.

V. Conclusions

Data sampling or different training data for different classifiers is considered to be the most effective technique for ensemble construction. Some of the popular ensemble methods manipulate available training patterns and some others induce generated new patterns to prepare a different training set for a particular classifier. This study investigates a new pattern generation method that modifies feature values of some original patterns with that of other patterns. The method is easy to understand and implement, and it is seem computationally economical. Extensive experiments have been carried out in this work to understand the proficiency of the proposed pattern generation method in ensemble construction. A suit of 30 benchmark classification problems was used for experimental studies. In many cases, it is found that the decision trees construction based on proposed pattern generation is beneficial for the performance of ensembles. The incorporation of proposed pattern generation method in bagging and RSM algorithms indicates that it can improve the performance of bagging and RSM algorithms.

References

[1] A. J. C. Sharkey, "On combining artificial neural nets," *Connection Science*, vol. 8, pp. 299-314, 1996.

[2] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorization," *Information Fusion*, vol. 6, pp. 99-111, 2005.

[3] J. Maudes, J. J. Rodriguez, C. Garcia-Osorio, and N. Garcia-Pedrajas, "Random feature weights for decision tree ensemble construction," *Information Fusion*, vol. 13, pp. 20-30, 2012.

[4] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 173-180, 2007.

[5] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.

[6] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. the 13th International Conf. on Machine Learning*, Morgan kaufmann, 1996, pp. 148-156.

[7] D. W. Opitz and R. Maclin, "Popular ensemble methods: an empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169-198, 1999.

[8] E. Bauter and R. Kohavi, "An empirical comparison of voting classification algorithms: bagging, boosting, and variants," *Machine Learning*, vol. 36, pp. 105-142, 1999.

[9] R. Polikar, "Bootstrap inspired techniques in computational intelligence," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 59–72, 2007.

[10] P. Melville, and R. J. Mooney, "Creating diversity in ensembles using artificial data," *Information Fusion*, vol. 6, pp. 99-111, 2005.

[11] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 832-844, 1998.

[12] G. Mart ńez-Muñoz and A. Suárez, "Switching class labels to generate classification ensembles," *Pattern Recognition*, vol. 38, pp. 1483–1494, 2005.

[13] C. Zhang and J. Zhang, "Rotboost-a technique for combining rotation forest and adaboost," *Pattern Recognition Letters*, vol. 29, pp. 1524-1536, 2008.

[14] S. Kotsiantis, "Combining bagging, boosting, rotation forest and random subspace methods," *Artif Intell Rev*, vol. 35, pp. 223–240, 2011.
[15] F. Bellal, H. Elghazel, and A. Aussem, "A semi-supervised feature ranking method with ensemble learning," *Pattern Recognition Letters*, vol. 33, pp. 1426-433, 2012.
[16] S. Bernard, S. Adam, and L. Heutte, "Dynamic random forests," *Pattern Recognition Letters*, vol. 33, pp. 1580-1586, 2012.
[17] D. Zhu, "A hybrid approach for efficient ensembles," *Decision Support Systems*, vol. 48, pp. 480–487, 2010.
[18] M. A. H. Akhand, M. M. Islam, and K. Murase, "A Comparative Study of Data Sampling Techniques for Constructing Neural Network Ensembles," *International Journal of Neural Systems*, vol. 19, no. 2, pp. 67-89, 2009.
[19] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, pp. 1399-1404, 1999.
[20] M. A. H. Akhand, and K. Murase, "Ensembles of Neural Networks based on the alteration of input feature values," *International Journal of Neural Systems*, vol. 22, pp. 77-87, 2012.
[21] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, *UCI Repository of Machine Learning Databases*, Dept. of Information and Computer Sciences, University of California, Irvine, 1998.
[22] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools with Java Implementations*, 2nd ed. Morgan Kaufmann, San Francisco, 2005.
[23] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, *Weka: Practical Machine Learning Tools and Techniques with Java Implementations*, Dept. of CS, University of Waikato, New Zeeland, 1999.
[24] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.
[25] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, pp. 83-98, 2005.
[26] A. Chandra, H. Chen, and X. Yao, "Trade-off between diversity and accuracy in ensemble generation," *Studies in Computational Intelligence (SCI)*, vol. 16, pp. 429–464, 2006.

M. A. H. Akhand received the B.Sc. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology (KUET), Bangladesh in 1999, the M.E. degree in Human and Artificial Intelligent Systems in 2006, and the doctoral degree in System Design Engineering in 2009 from University of Fukui, Japan. He joined as a lecturer at the Department of Computer Science and Engineering at KUET in 2001, and is now an associate professor. He is a member of Institution of Engineers, Bangladesh (IEB). His research interest includes artificial neural networks, evolutionary computation, swarm intelligence and other bio-inspired computing techniques.

M. M. Hafizur Rahman received his B.Sc. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology (KUET), Khulna, Bangladesh, in 1996. He received his M.Sc. and Ph.D. degree in Information Science from the Japan Advanced Institute of Science and Technology (JAIST) in 2003 and 2006, respectively. Rahman is now an assistant professor in the Dept. of Computer Science, KICT, IIUM, Malaysia. Prior to join in the IIUM, he was an associate professor in the Dept. of CSE, KUET, Khulna, Bangladesh. He was also a visiting researcher in the School of Information Science at JAIST and a JSPS postdoctoral research fellow at Research Center for Advanced Computing Infrastructure, JAIST & Graduate School of Information Science (GSIS), Tohoku University, Japan in 2008 and 2009 & 2010-2011, respectively. His current researches include parallel and distributed computer architecture, hierarchical interconnection networks, and optical switching networks. Rahman is a member of IEICE of Japan and IEB of Bangladesh.

K. Murase is a professor at the Graduate School of Engineering, University of Fukui, Fukui, Japan, since 1999. He received M.E. in Electrical Engineering from Nagoya University in 1978, Ph.D. in Biomedical Engineering from Iowa State University in 1983. He Joined as a research associate at Department of Information Science of Toyohashi University of Technology in 1984, and as an associate professor at the Department of Information Science of Fukui University in 1988. He became the professor in 1992. He is a member of The Institute of Electronics, Information and Communication Engineers (IEICE), The Japanese Society for Medical and Biological Engineering (JSMBE), The Japan Neuroscience Society (JSN), The International Neural Network Society (INNS), and The Society for Neuroscience (SFN). He serves as a board of directors in Japan Neural Network Society (JNNS), a councilor of Physiological Society of Japan (PSJ) and a councilor of Japanese Association for the Study of Pain (JASP).