Diversity-Based Boosting Algorithm

Jafar A. Alzubi
School of Engineering
Al-Balqa Applied University
Al-Salt, Jordan

***Abstract*—Boosting is a well known and efficient technique for constructing a classifier ensemble. An ensemble is built incrementally by altering the distribution of training data set and forcing learners to focus on misclassification errors. In this paper, an improvement to Boosting algorithm called DivBoosting algorithm is proposed and studied. Experiments on several data sets are conducted on both Boosting and DivBoosting. The experimental results show that DivBoosting is a promising method for ensemble pruning. We believe that it has many advantages over traditional boosting method because its mechanism is not solely based on selecting the most accurate base classifiers but also based on selecting the most diverse set of classifiers.**

Keywords*—Artificial Intelligence; Classification; Boosting; Diversity; Game Theory.

I. Introduction

Boosting is a powerful mechanism for improving the performance of classifiers that has been proven to be theoretically and empirically sound. The ADAptive BOOSTing (AdaBoost) algorithm, developed by Freund and Schapire [1] in 1995, has shown remarkable performance on solving benchmark real world problems, and it is been recognized as the best "off-the-shelf" learning algorithm.

The idea of the algorithm is to use a weak learner (e.g., decision tree) as the base classifier to be boosted. However, just like any other ensemble learning design, AdaBoost builds a composite hypothesis by combining many individual hypotheses through a weighted voting mechanism. Unfortunately, in many tasks and for the sake of reaching a reasonable accuracy, the number of base classifiers must be increased. It is obvious that enlargement in the design requires a huge amount of memory to store these hypotheses [2]. In fact, this requirement makes such ensemble method impractical to be deployed in many real applications. This drawback came to the attention of researchers in the machine learning field, prompting many solutions to be proposed [3][4][5]. One of the early suggestions called for an empirical pruning technique called Kappa pruning method to perform pruning on the boosting ensemble constructed of decision trees [4][6][7]. Their objective was to accomplish the task while maintaining the accuracy rate.

This work proposes a potential improvement to the AdaBoost by applying Coalition Based Ensemble Design algorithm (CED) [8][9] to be an intermediate phase in AdaBoost. Although the problem of pruning the boosting algorithm is intractable and hard to approximate, This work suggests a *margin*-based heuristic approach for solving this problem.

II. Related Work

This section presents a review of the methods that have been introduced to prune boosting algorithms. Pruning techniques described in the literature can be classified into two main categories: first, techniques that combine sub-ensembles based on the error rate estimated on the training set. The second, techniques which use some of the diversity measures, in particular the pair-wise measures, to build the subset of classifiers [4][10][11].

However, the first category is not very effective in producing a better sub-ensemble than the whole ensemble. As in the case of boosting, the generated classifiers are typically driven to zero training error very quickly [12]. Therefore, sub-ensembles based on this approach are similar and it is not easy to distinguish between them.

In 1997 Margineantu and Dietterich [4] were the first who studied the problem of pruning boosting algorithm and in particular AdaBoost. They presented five pruning methods: Early stopping, K-L Divergence pruning, Kappa pruning, Kappa-error convex hull pruning, and Reduce-error pruning with back fitting.

Later, Tamon and Xiang [5] suggested a modification to the Kappa Pruning method proposed by Margineantu and Dietterich [4]. They introduce what is called "weight shifting" strategy as an alternative heuristic approach to Kappa pruning. They further explained that while the voting weight of pruned hypothesis in kappa pruning assigned zero, in their proposed method it transfers that voting weight to the unpruned hypothesis.

The process by which this weight is transferred is based on measuring the similarity between the pruned hypothesis and the rest of the unpruned hypotheses, where each of them will receive a fraction of the weight proportional to its distance from the pruned hypothesis. The closer an unpruned hypothesis to prune one the higher its share of the distributed weight will be. This weight allocation mechanism has been called soft assignment according to [5][13], who claimed that it yields more faithful final ensemble, especially when a high pruning rate is required.

Hernandez-Lobato et al. [11] in 2006 presented a completely different heuristic approach for pruning AdaBoost which is based on application of a genetic algorithm. They defined the base classifiers that are returned by AdaBoost as their population. The fitness function is the created ensemble accuracy, and the optimization problem is to find the best subset of a given set of classifiers.

In [11] they conclude that the results of experiments which carried over a variety of domains support their claim that the genetic algorithm outperforms or is at least as good as the heuristic methods that have been used such as Kappa pruning and Reduce-error pruning which they compared their work with.

To avoid the drawbacks of the methods used in literature, we introduced our algorithm CED (Figure 2) which is based on calculating the contribution of diversity for each one of the classifiers in the ensemble and create a coalition based on these calculations which later will construct the sub-ensemble.

III. Design of DivBoosting Algorithm

AdaBoost is one of the most powerful and successful ensemble methods. It shows an outstanding performance in many classification tasks and it outperforms bagging in many situations. The drawback of AdaBoost is that it suffers and seriously deteriorates if there is a noise in the class labels. This disadvantage occurs because of the weight adaptation nature of the algorithm that it applies it on the training data set.

Here we shall present our improved version of AdaBoost algorithm which we called Diverse Boosting ($DivBoosting$). Figure 1 shows the flow chart of $DivBoosting$ and full details of the algorithm functionality are presented as a pseudo code in algorithm 1. It is worth mentioning here, that the implementation of AdaBoost we are considering here is the $resampling$ version.

$DivBoosting$ is an iterative algorithm. It starts by initializing the set of candidate classifiers to an empty set, then starts its training process by assigning uniform weights, w^0 on D_{trn}. Then it proceed with the following loop: generates a bootstrap sample S_k using the weight w^k. Create the classifier e_k using the sample S_k for training.

The next major step is to calculate ϵ_k which represents the weighted error for e_k on D_{trn} using the set of weights w^k. In contrast to other versions of AdaBoost, the algorithm does not stop if either of the two conditions is met; first, if the error ϵ_k is equal to zero and second if the error ϵ_k is equal to or greater than 0.5. Instead the weights w^{k+1} are reset to uniform values and process repeated. In case of the error ϵ_k is greater than zero and less than 0.5 then a new weights are calculated. This loop stops when the desire number of base classifiers is generated.

The previous iterative process produces a set of candidate base classifiers that form the input for the next phase. The subroutine $Excecuting_C ED$ execute the CED algorithm that is explained in details in [8]. $DivBoosting$ uses weighted majority vote as a combining method.

The final output of $DivBoosting$ is an optimal ensemble composed of base classifiers that are complementary (diverse) which means their errors are uncorrelated. The objective of $DivBoosting$ is to produce an ensemble that outperforms the original ensemble in term of both accuracy and ensemble size.

IV. Experimental Data

To verify our theoretical assertion that the DivBoosting algorithm will have an improvement in performance over the conventional AdaBoost algorithm, and further illustrate how DivBoosting works, several experiments conducted on nine real data sets from the UCI repository [14]. In addition to one experiment performed on the blog spam data set - a data set we built- in order to see the effect of DivBoosting on large data set with a large number of features. Table I summarizes

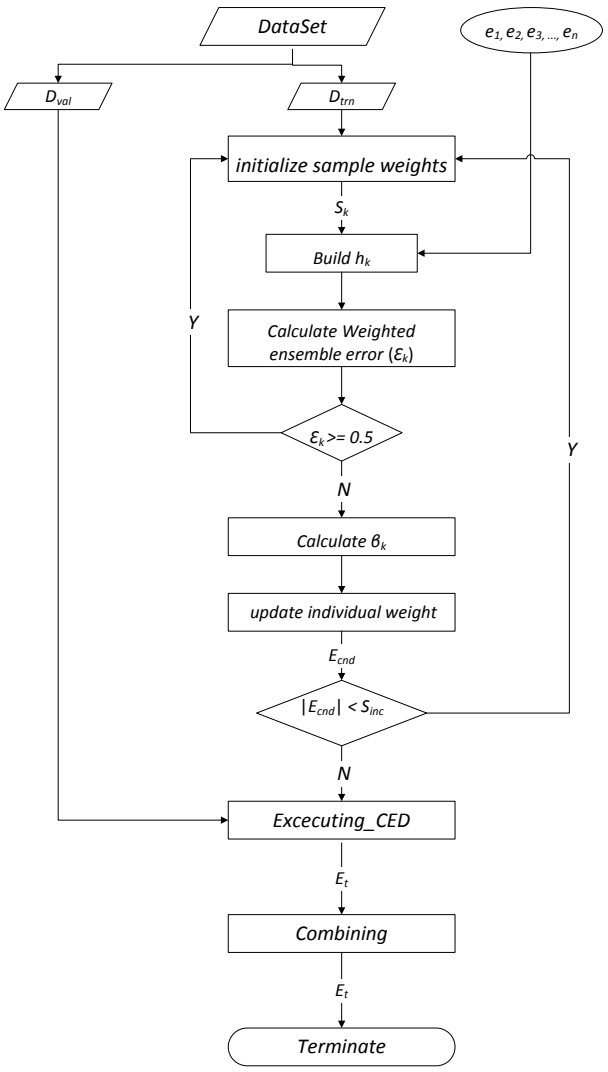

Fig. 1: Flowchart of Diverse Boosting algorithm (DivBoosting)

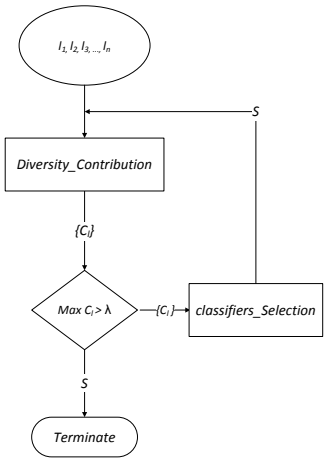

Fig. 2: Flow chart of Coalition Based Ensemble Design algorithm (CED)

Algorithm 1 pseudo code of Diverse Boosting algorithm (DivBoosting)

Input:

D_{trn} : *training data*
D_{val} : *validation data*
ζ : *a training algorithm*
E_{cnd} : *set of incremental cadidate classifiers*
S_{inc} : *size of the incremental candidate classifiers set*
η : *a combining method*

Output:

E_t : *target ensemble*

Set weights $w_j^1 = \frac{1}{N}$ // weights $w_1 = [w_1, ..., w_N], w_j^1 \in [0,1], \sum_{j=1}^{N} w_j^1 = 1$
$E_{cnd} := \phi$ // Initialize the candidate classifiers set
for $k = 1, ..., S_{inc}$ **do**
  $S_k = sample\ from\ D_{trn}\ using\ distribution\ w^k$
  $e_k = Training(\zeta, S_k)$; // Create a learning base classifier
  Calculate weighted ensemble error at step k :

$$\epsilon_k = \sum_{j=1}^{N} w_j^k l_k^j \tag{1}$$

  $(l_j^k = 1\ if\ e_k(x_j) \neq y_j\ and\ l_j^k = 0\ if\ e_k(x_j) = y_j)$
  if $\epsilon_k = 0 \mid \epsilon_k \geq 0.5$ **then**
    ignore D_k
    $w_j^k = \frac{1}{N}$
  else
    Calculate

$$\beta_k = \frac{\epsilon_k}{1 - \epsilon_k} \quad , where\ \epsilon_k \in (0, 0.5) \tag{2}$$

    Update individual weights

$$w_j^{k+1} = \frac{w_j^k \beta_k^{(1-l_k^j)}}{\sum_{i=1}^{N} w_i^k \beta_k^{(1-l_k^j)}} \quad , j = 1, ..., N \tag{3}$$

  end if
  $E_{cnd} = E_{cnd} \bigcup e_k$
end for
$E = Excecute\_CED(E_{cnd}, D_{val})$; // Execute CED algorithm 2
$E_t = Combining(E, \eta)$;
return E_t

the used data sets in terms of number of examples, features, and classes.

The ensembles that used in the experiments were homogeneous ensembles which means the base classifiers were all the same (100 C4.5 decision trees). The performance of each decision tree was evaluated using five complete runs of five fold cross validation. In each five-fold cross-validation, each data set is randomly split into five equal size partitions and the results are averaged over five trails. In this case, one partition is set aside for testing, while the remaining data is available for training.
To test the performance on varying ensemble sizes, learning curves were generated by the system after forming sub-ensembles with different sizes (20%, 30%, 40%, 50%, 60%, 80%, and 100%). The sub-ensemble sizes of the generated ensemble represented as points on the learning curve.

TABLE I: Summary of Data Sets

Name	Examples	Classes	Attributes
Blog Spam	56000	2	547
Breast Cancer	699	2	9
Letter Recognition	20000	26	16
Iris	150	3	4
Segment	2310	7	19
ionosphere	351	2	34
Statlog (Vehicle Silhouettes)	946	4	18
Haberman's Survival	946	2	3
Contraceptive Method Choice	1473	2	3
Isolet	1559	26	617
glass	214	6	9
colic	368	2	22
heart-c	303	2	13
splice	3190	3	62
Anneal	898	6	38

For the purpose of comparing DivBoosting with AdaBoost across all domains we implemented statistics used in [15][16], specifically the win/draw/loss record and the geometric mean error ratio. The simple win/draw/loss record computed by calculating the number of data sets for which DivBoosting obtained better, equal, or worse performance than Boosting with respect to the ensemble classification accuracy. In addition to that, we computed another record representing the *statistically significant* win/draw/loss, according to this record win/loss is only computed if the difference between two values is greater than 0.05 level which was determined to be significant by computing the student paired *t*-test.

V. Experimental Results

Our results are summarized in Table II. Each cell in this table presents the accuracy of DivBoosting versus AdaBoost algorithm. We varied the sub-ensembles sizes from 20% to 100% of the generated ensemble, with more points lower on the learning curve because this is where we expect the difference to be the most between the two algorithms. A summary of the statistics is presented at the bottom of the table for each point on the learning curve.

For a better visualization of the results presented in the

TABLE II: Accuracy Rate of DivBoosting VS. AdaBoost using homogeneous ensembles

Classifiers	20%	30%	40%	50%	60%	80%	100%
Blog Spam	97.84/95.33	95.70/95.69	97.69/96.32	97.40/97.06	97.40/97.21	97.40/96.96	96.82/96.82
Breast Cancer	95.38/93.52	96.49/93.51	95.67/93.95	95.67/94.18	95.23/94.18	95.67/94.83	95.59/95.59
Ionosphere	92.20/87.14	93.36/87.89	93.36/87.90	92.71/88.97	91.21/89.56	91.21/89.66	89.64/89.64
Iris	87.69/81.73	94.38/88.06	94.61/88.11	94.01/91.97	91.50/90.85	91.20/91.00	91.66/91.66
splice	66.12/56.74	73.28/59.10	77.32/60.11	72.55/60.25	68.04/61.14	60.82/61.36	59.91/59.91
Colic	76.16/73.14	81.17/75.32	82.67/77.29	81.86/78.46	80.74/78.84	79.96/80.23	80.76/80.76
Haberman's	71.43/65.01	71.94/66.21	70.49/68.30	71.16/68.44	69.80/68.99	66.50/67.10	65.89/65.89
Auto (Statlog)	66.87/58.77	73.15/61.23	74.54/64.33	72.45/67.37	73.19/68.46	71.08/67.94	69.78/69.78
Heart-c	71.85/68.74	76.27/72.15	76.55/71.40	75.63/73.28	78.21/74.34	75.81/73.90	76.29/76.29
Letter Recognition	75.41/70.21	79.12/73.50	83.10/78.15	82.76/79.81	80.51/78.60	81.31/77.09	78.21/78.21
Win/Draw/Loss	15/0/0	15/0/0	15/0/0	15/0/0	15/0/	15/0/0	0/15/0
Sig. W/D/L	15/0/0	15/0/0	15/0/0	15/0/0	15/0/0	15/0/0	0/15/0
GM error ratio	0.7345	0.6963	0.6777	0.8043	0.8824	0.9180	1.0000

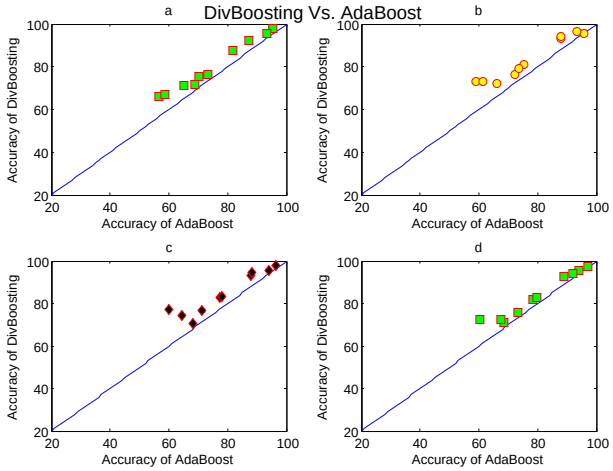

Fig. 3: A comparison showing the DivBoosting versus AdaBoost algorithm on all data sets given various homogeneous ensembles: a) 20 classifiers b) 30 classifiers c) 40 classifiers d) 50 classifiers

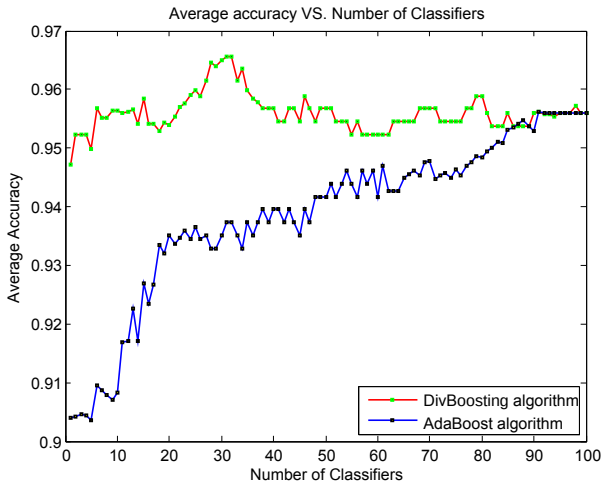

Fig. 4: Learning curve showing the average accuracy versus the number of classifiers produced by both DivBoosting and AdaBoost algorithms on breast cancer data set using 30% of data for training and a homogeneous ensemble.

table, we present plots in figures 4 to 6. Each plot present a comparison of DivBoosting versus AdaBoost for one data set over all sub-ensemble sizes from 1% to 100% of the generated ensemble.

The results in Table II confirm our assumption that combining the predictions of DivBoosting ensembles will, on average, have accuracy improvement over the AdaBoost. According to this table, we have the following general observations: 1) DivBoosting algorithm can generally improve the classification performance across all domains. 2) the best gain in performance is achieved when the ensemble accuracy of the data set is low.

For the results in Table II which represents the homogeneous ensembles, DivBoosting has more significant wins to

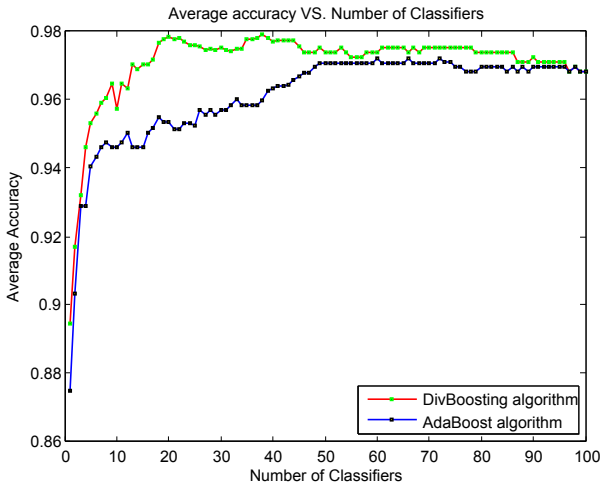

Fig. 5: Learning curve showing the average accuracy versus the number of classifiers produced by both DivBoosting and AdaBoost algorithms on Blog spam data set using 30% of data for training and a homogeneous ensemble.

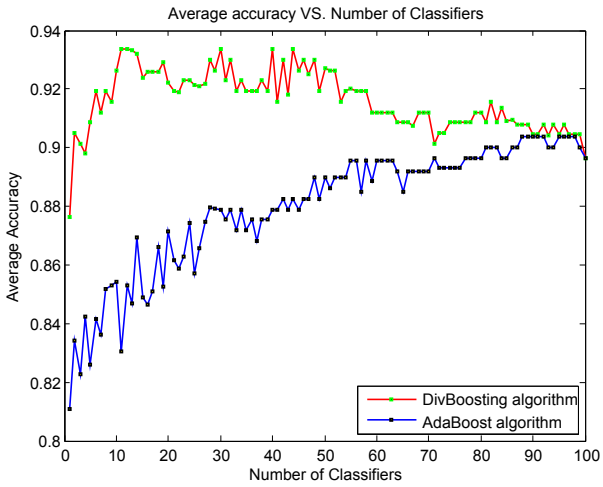

Fig. 6: Learning curve showing the average accuracy versus the number of classifiers produced by both DivBoosting and AdaBoost algorithms on Ionosphere data set using 40% of data for training and a homogeneous ensemble.

losses over AdaBoost for all data points along the learning curve. DivBoosting also outperforms AdaBoost on the geometric error ratio. This suggests that even in cases where gain is not achieved no loss occurred at any point.

We produce a scatter plots in figures 3 for various sub-ensembles sizes from Tables II. Figure 3 shows the homogeneous sub-ensembles case where DivBagging outperforms AdaBoost algorithm, in particular in sub graph c case.

DivBoosting outperforms AdaBoost early on the learning curves both on significant wins/draw/loss and geometric mean ratio. However, the trend becomes less obvious when the ensemble size increases and getting closer to the maximum size (consisting of all base classifiers). Note that even with large ensemble size, DivBoosting performance is quite competitive with AdaBoost, given ensemble sizes of 80% to 95% base classifiers, DivBoosting produces higher accuracies on all data sets with all training data set sizes.

On all data sets, DivBoosting achieves a higher accuracy rate than AdaBoost with less ensemble size. Figures 4 to 6 show learning curves that clearly demonstrate this point. To determine the influence of DivBoosting algorithm on the ensemble size, we chose to present a comparison of accuracy versus ensemble size for DivBoosting and AdaBoost on three data sets (see figures 4 to 6). The performance on other data sets is similar. We note, in general, that the accuracy of AdaBoost increases with ensemble size while the accuracy of DivBoosting increases when the diversity of the ensemble increases. So on most data sets, the performance reach its highest level when the ensemble size is between 20% and 30% of the generated ensemble size.

Figure 4 shows the performance of both algorithms on breast cancer data set for homogeneous ensembles. DivBoosting achieves an accuracy rate of 96.55% with ensemble size of 31 where AdaBoost's highest accuracy of 94.62% occurred at ensemble size of 91. These results yield a reduction of 65.93% in the ensemble size and a gain of 3% in the accuracy at the same size level.

The curve of Ionosphere data set in figure 6 illustrates that DivBoosting reaches an accuracy rate of 93.37% with ensemble size of 12 comparing to AdaBoost which it achieved an accuracy of 90.36% at ensemble size of 88. So the reduction in size here is 86.4% and at the same time the accuracy increased by 9.59%. A similar pattern observed on the Contraceptive Method Choice data set where a reduction of 64.56% in the ensemble size and 11.59% increases in the accuracy obtained. The learning curve of Blog Spam data set in figure 5 demonstrates the performance of DivBoosting and AdaBoost on a large data set in terms of both number of examples and number of features. Apart from a 2.63% improvement in the accuracy which is not small when taken in relation to the high performance of AdaBoost on this data set, the trend of ensemble reduction is the same as with other data sets which is 66.66%.

VI. Conclusions

DivBoosting is a very powerfull and effective algorithm to increase the classification accuracy and reduce the ensemble size. Throughout this paper, we introduced the algorithm and evaluated its performance through extensive experiments in comparison with conventional AdaBoost algorithm. We conducted a set of experiments using homogenous ensembles where the base learners are decision trees. DivBoosting shows the ability to increase the classification accuracy and achieves a lower ensemble size than AdaBoost.

The experimental results show that DivBoosting achieves significant improvements over AdaBoost in all domains, and yet reduces the ensemble size with more than 40% compared to the one produced by AdaBoost. Generally speaking, DivBoosting is a promising ensemble algorithm that inherits the efficiency of AdaBoost and the size reduction of CED algorithm.

References

[1] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, EuroCOLT '95

Proceedings of the Second European Conference on Computational Learning Theory, London, UK, (1995) 23-37.

[2] O. A. Alzubi, J. A. Alzubi, S. Tedmori, H. Rashaideh, and O. Almomani, *Consensus-Based Combining Method for Classifier Ensembles*, International Arab Journal of Information Technology, 15(2), (2018).

[3] G. Martinez-Munoz and A. Suarez, *Using Boosting to Prune Bagging Ensembles*, Pattern Recognition Letters, 28, (2007) 156-165.

[4] D. Margineantu and G. T. Dietterich, *Pruning Adaptive Boosting*, ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning, San Francisco, CA, USA, (1997) 211-218.

[5] C. Tamon and J. Xiang, *On the Boosting Pruning Problem*, ECML '00 Proceedings of the 11th European Conference on Machine Learning, London, UK, (2000) 404-412.

[6] T. Dietterich and D. Fisher, *An experimental comparison of three methods for constructing ensembles of decision trees, in Bagging, boosting, and randomization*, Machine Learning, (2000) 139-157.

[7] S. Shylaja, K. Balasubramanya, S. Natarajan, R. Muthuraj, and S. Ajay, *Feed Forward Neural Network Based Eye Localization and Recognition Using Hough Transform*, International Journal of Advanced Computer Science and Applications, 2(3), (2011).

[8] J. A. Alzubi, *Optimal Classifier Ensemble Design Based on Cooperative Game Theory*, Research Journal of Applied Sciences, Engineering and Technology, 11(12), (2015) 1336-1343.

[9] J. A. Alzubi, *Diversity Based Improved Bagging Algorithm*, ICEMIS '15 Proceedings of the The International Conference on Engineering and MIS, Istanbul, Turkey, (2015).

[10] Y. Zhang, S. Burer, W. Street, K. Bennett, and E. Parrado-hern, *Ensemble Pruning Via Semi-definite Programming*, Journal of Machine Learning Research, 7, (2006) 1315-1338.

[11] D. Hernández-Lobato, J. Hernández-Lobato, R. Ruiz-Torrubiano, and A. Valle*Pruning adaptive boosting ensembles by means of a genetic algorithm*, IDEAL'06 Proceedings of the 7th international conference on Intelligent Data Engineering and Automated Learning, Burgos, Spain, (2006) 322-329.

[12] R. Schapire, and Y. Freund, *Boosting the margin: a new explanation for the effectiveness of voting methods*, The Annals of Statistics, 26, (1998) 322-330.

[13] D. Opitz and R. Maclin, *Popular ensemble methods: An empirical study*, Journal of Artifficial Intelligence Research, 11, (1999) 169-198.

[14] D. Newman, S. Hettich, C. Blake, and C. Merz, *UCI Repository of machine learning databases*, University of California, Irvine, Dept. of Information and Computer Sciences, (1998), http://www.ics.uci.edu/ mlearn/MLRepository.html.

[15] G. I. Webb, *MultiBoosting: A Technique for Combining Boosting and Wagging*, Machine Learning, 40(2), (2000) 159-196.

[16] E. Bauer and R. Kohavi, *An empirical comparison of voting classification algorithms: Bagging, boosting, and variants*, Machine Learning, (1998) 105-139.