On Adjustment Functions for Weight-Adjusted Voting-Based Ensembles of Classifiers

Kuo-Wei Hsu
Department of Computer Science, National Chengchi University, Taipei, Taiwan
Center for Computational Research and Applications, National Chengchi University, Taipei, Taiwan
Email: hsu@cs.nccu.edu.tw

***Abstract*—An ensemble of classifiers is a system consisting of multiple member classifiers which are trained individually and whose outcomes are aggregated into an overall outcome for a testing data instance. Voting is a common approach used to aggregate outcomes generated by member classifiers. Ensembles based on weighted voting have been studied for some time. However, the focus of most studies is more on weight assignment rather than on weight adjustment, whose basic idea is to increase the weights of votes from member classifiers performing better on data instances of higher difficulty. In this paper, we present our study on adjustment functions in each of which both the performance of a member classifier and the difficulty of a data set are determined nonlinearly. We report results from experiments conducted on several data sets, demonstrating the potential of the studied functions.**

***Index Terms*—Classification, ensemble, voting**

I. Introduction

A classifier is trained with a given set of data instances, or a given data set; after training, it generates an outcome or a result of classification for a testing data instance. An ensemble of classifiers, or a multiple classifier system, is a system that consists of multiple member classifiers which are trained individually and whose outcomes are aggregated into an overall outcome for a testing data instance [1], [2]. It is a form of collective intelligence and has caused general interests. Using an ensemble is advantageous in that a group of classifiers working together is usually superior to individual classifiers working alone [3].

For training member classifiers that will be used to construct an ensemble, a commonly used approach is to perform sampling on a given data set so as to make member classifiers different. Because using a group of multiple identical member classifiers is the same as using just one of them and is not favorable from the point of view of collective intelligence, member classifiers are expected to be different so that they would generate diverse outcomes. For example, bootstrap aggregating, or bagging for short, uses the bootstrap procedure to perform sampling on a given data set when training member classifiers that will be used to construct an ensemble [4], [5]. For using an ensemble to generate an outcome for a testing data instance, a common approach is to use voting to aggregate outcomes generated by member classifiers [6]. For example, bagging uses voting to aggregate outcomes [4], [5]. Despite its simplicity, voting has been found effective in practice and has been applied in various applications, such as character recognition [7], spam filtering and intrusion detection [8].

Intuitively, after member classifiers are trained, one can extend the simple voting process by assigning a weight to votes from each member classifier according to its performance, such as accuracy [9]. One can find situations where the weighted voting process is better than the simple voting process [10]. In fact, ensembles based on weighted voting have been studied for some time [11]. However, the focus of most studies is more on weight assignment [11] rather than on weight adjustment [12]; the basic idea of the former is to assign higher weights to votes from member classifiers that perform better, while the basic idea of the latter is to increase the weights of votes from member classifiers performing better on data sets that are more difficult to be classified correctly.

In this paper, we present our study on functions used in weight adjustment. The idea of using weighted voting to aggregate outcomes generated by member classifiers is not new, and neither is the idea of using weight adjustment. This paper is different from others because in each of the studied functions both the performance of a member classifier and the difficulty of a data set are determined nonlinearly. Considering the reproducibility of this study, for evaluation, we extended an open-source machine learning package and conducted experiments on several data sets available on the Internet. We report the experimental results and demonstrate the potential of the studied functions.

The rest of this paper is organized as follows: We discuss related work in Section II, describe ensemble construction in Section III, report experimental results in Section IV, and give conclusions in Section V.

II. Related Work

The weighted voting process extends the simple voting process by counting weights of votes from member classifiers of an ensemble. It has been used in many areas.

Manuscript received July 30, 2013; revised August 30, 2013; accepted September 25, 2013.

doi:10.4304/jcp.9.7.1547-1552

For example, Stepenosky et al. proposed to use the weighted combination rules to construct ensembles, targeting to help fight Alzheimer's disease [13]. The weight of votes from a member classifier of an ensemble usually depends on the performance achieved by the member classifier [9]. Bella et al. considered the posterior probabilities associated with outcomes generated by classifiers, and they presented a study on probability calibration and classifier combination [14].

Littlestone and Warmuth offered a different view of the weighted voting process, using results of serial trials to set weights [15]. Tsoumakas, Katakis, and Vlahavas stated that the simple (unweighted) voting process and the weighted voting process are two of the simplest approaches used in classifier combination and both work for homogeneous member classifiers (trained by the same algorithm) and heterogeneous member classifiers (trained by different algorithms) [16].

Moreno-Seco et al. presented linear as well as nonlinear functions to assign weights to votes from member classifiers of an ensemble and they proposed three methods for weighted voting [17]: First, RSWV (Re-Scaled Weighted Vote), whose basic idea is to assign no weights to votes from any member classifier in which the number of training errors is larger than a threshold. Second, BWWV (Best-Worst Weighted Vote), whose basic idea is to set the weight for the best member classifier to 1, set the weight for the worst member classifier to 0, and have the weights for other member classifiers linearly proportional to their training performance. Third, QBWWV (Quadratic best-worst weighted vote), whose basic idea is similar to the basic idea of BWWV but different in that it uses a quadratic function taking the training performance as the input to set weights for member classifiers other than the best and the worst. Major differences between their work and our work are as follows: First, their functions are used in weight assignment instead of weight adjustment. Second, their functions do not consider the difficulties of data sets used in training.

Some researchers proposed to use numerical optimization techniques in weight assignment. For example, He, Yang, and Kong used a genetic algorithm to determine weights of support vector machine based member classifiers of an ensemble [18]. For training neural network based member classifiers to construct an ensemble, Shen and Kong used a genetic algorithm to determine weights that would be given to votes from these member classifiers [19], while Chen and Yu used a particle swarm optimization based method to determine weights [20].

Kim et al. proposed an approach to weight adjustment whose basic idea is to decrease the biases caused by the different quality levels of the data sets used to train member classifiers [12]. They used two weight vectors: One is instance weight vector, which represents the difficulty of a data set; the more difficult a data set tends to be classified correctly, the higher weight the data set will receive. The other is classifier weight vector, which represents the effectiveness of each member classifier of an ensemble and is used in the weighted voting process; the better a member classifier is, the better performance it will achieve, and the higher weight it will receive. Furthermore, the content of one vector depends on and recursively calculated by the content of the other vector: The weight of a data set is higher as more good member classifiers make errors on it, and the weight of a member classifier is higher as it correctly classifies more difficult data sets (as it correctly classifies more data instances in data sets that are difficult and/or as it correctly classifies data instances in data sets that are more difficult). Kim et al. presented an iterative procedure to obtain the optimal values of these two vectors, and they proved that these two vectors can be obtained with eigenvectors [12]. The basic idea of the weight adjustment functions studied in this paper is similar to that of the approach proposed by Kim et al., but there are differences between our work and theirs: First, in our work, the influence of the weights of data sets on the weights of member classifiers is nonlinear, and vice versa. Second, we adjust weights iteratively because the computational complexity of finding eigenvectors is high. Third, Kim et al. assume that data sets are of good quality but we do not make such an assumption.

Moreover, some researchers proposed to dynamically update weights of member classifiers of an ensemble that adopts weighted voting [21], [22]; however, they worked on the setting for incremental learning, while the setting of our study is for batch learning.

III. Ensemble Construction

Figure 1 presents the pseudo code of our ensemble construction procedure. It is in a C-like style, and so are the following procedures. Additionally, in Figures 1-3, names of functions are in bold text and names of constant variables are in italic text.

In brief, the procedure presented in Figure 1 works as follows: It trains a number of member classifiers, assign weights to them, and finally adjusts weights to them; a weight of a member classifier will be used as the weight assigned to votes from the member classifier. In order to train a member classifier, by using the bootstrap procedure to perform sampling on the data set, the procedure generates a data set (which would be different from the data sets used in training other member classifiers) that will be given to an algorithm that builds a classifier in a systematic way.

```
0   ConstructEnsemble() {
1     initialize variables;
2     for (i = 1; i <= NUM_MEMBER_CLASSIFIERS; i++) {
3       tr_data = DoBootstrapSampling(data);
4       mem_classifiers[i] = BuildClassifier(tr_data);
5       if UNSAMPLED_DATA_FOR_EVALUATION is set to true
6         eval_datv[i] = GetUnsampledData(data, tr_data);
7       else
8         eval_datv[i] = tr_data;
9     }
10    AssignWeights();
11    AdjustWeights();
12  }
```

Figure 1. Pseudo code of our ensemble construction procedure.

Figure 2 presents the pseudo code of the weight assignment procedure used in our ensemble construction procedure. Accuracy and F1-measure are common options for weight assignment. In Figures 2 and 3, the array v[][] holds elements whose values are evaluation results measured in accuracy or F1-measure. Accuracy or F1-measure is often calculated by referring to labels of sampled data instances (used in training). However, the performance that a classifier achieves on the data set used in its training may not be an ideal indicator to its performance on new and unseen data instances. That is, assigning weights to votes of member classifiers according to how they performed in training may increase the risk of overestimating what they will achieve. The risk could be reduced by using unsampled data instances (not used in training) to evaluate member classifiers.

```
0   AssignWeights() {
1     initialize variables;
2     for (int i=1; i <= NUM_MEMBER_CLASSIFIERS; i++) {
3       for (int j=1; j <= NUM_MEMBER_CLASSIFIERS; j++) {
4         eval_results = Evaluate(mem_classifiers[i], eval_datv[j]);
5         switch (EVALUATION_METRIC) {
6           case ACCURACY:
7             v[i][j] = CalculateAccuracy(eval_results);
8           case F1MEASURE:
9             v[i][j] = CalculateF1Measure(eval_results);
10        }
11        if i is equal to j
12          mem_weights[i] = v[i][j];
13      }
14    }
15  }
```

Figure 2. Pseudo code of the weight assignment procedure used in our ensemble construction procedure.

Figure 3 presents the pseudo code of the weight adjustment procedure used in our ensemble construction procedure. For an ensemble, the process that it uses to aggregate outcomes generated by its member classifiers related to its overall performance, and so are the algorithm and the data sets used to train its member classifiers. Here we use weighted voting, and we focus more on weight adjustment rather than on weight assignment. When adjusting weights, we consider both the performance of each member classifier and the difficulty of each data set.

In Figure 3, E is the Euler's number; p[] is the array of the relative performance of each member classifier, and d[] is the array of the relative difficulty of each data set used in training or evaluation; fp and fd are coefficients associated with the relative performance and the relative difficulty, respectively. The functions used in weight adjustment are based on logarithmic and exponential functions. They are illustrated in Figure 4, where the x-axis is evaluation result obtained in weight assignment and the y-axis is fp or fd. They determine nonlinearly both the performance of a member classifier and the difficulty of a data set. Such nonlinearity is what makes the weight assignment procedure presented in Figure 3 different from the approach proposed in Ref. [12].

```
0   AdjustWeights() {
1     initialize variables;
2     ArrayCopy(mem_weights, p);
3     while ( t <= MAX_ITERATIONS) {
4       for (int j=1; j <= NUM_MEMBER_CLASSIFIERS; j++) {
5         for (int i=1; i <= NUM_MEMBER_CLASSIFIERS; i++) {
6           switch (FUNCTION) {
7             case LOGARITHMIC: fp = Log2(2-v[i][j]);
8             case EXPONENTIAL: fp = (Exp(1-v[i][j])-1)/(E-1);
9           }
10          d[j] = d[j] + fp*p[i];
11        }
12        d[j] = d[j]/NUM_MEMBER_CLASSIFIERS;
13      }
14      for (int i=1; i <= NUM_MEMBER_CLASSIFIERS; i++) {
15        for (int j=1; j <= NUM_MEMBER_CLASSIFIERS; j++) {
16          switch (FUNCTION) {
17            case LOGARITHMIC: fd = Log2(1+v[i][j]);
18            case EXPONENTIAL: fd = (Exp(v[i][j])-1)/(E-1);
19          }
20          p[i] = p[i] + fd*d[j];
21        }
22        p[i] = p[i]/NUM_MEMBER_CLASSIFIERS;
23        mem_weights[i] = p[i];
24      }
25      t++;
26    }
27  }
```

Figure 3. Pseudo code of the weight adjustment procedure used in our ensemble construction procedure.

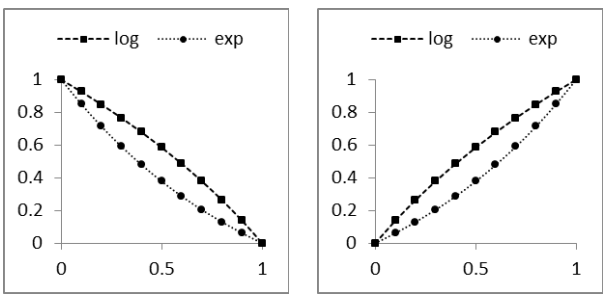

Figure 4. Coefficients fp (left) and fd (right) against evaluation results obtained in weight assignment.

The basic idea of the above weight adjustment functions can be divided into the following points:

- As more member classifiers perform poorly on a data set, the data set would be more difficult.

Among the member classifiers performing poorly on the data set, as more are classifiers that show good performance in general, we are more confident that the data set is a difficult data set.

- As more member classifiers perform well on a data set, the data set would be easier. Among the member classifiers performing well on the data set, as more are classifiers that do not show good performance in general, we are more confident that the data set is an easy data set.
- As a classifier performs well on more data sets, the classifier is a good classifier. Among the data sets on which the classifier performs well, as more are difficult data sets, we are more confident that the classifier is a good classifier.
- As a classifier performs poorly on more data sets, the classifier is not a good classifier. Among the data sets on which the classifier performs poorly, as more are easy data sets, we are more confident that the classifier is not a good classifier.

Of course, there are other possible functions for weight adjustment, and part of our future work is to investigate other possible functions that follow the above points.

Last but not least, what is presented in Figure 3 is an iterative procedure, while we can use an additional condition, such as a condition for convergence, to stop the procedure early.

IV. Experimental Results

Data sets used in experiments were downloaded from the Internet [23], [24]. They are binary data sets and summarized in Table I, where the first (leftmost) column is for their names, the second column is for their numbers of data instances, the third column is for their numbers of attributes, and the fourth (rightmost) column is for their percentages of the minority class. The rows are sorted by the number of data instances in ascending order.

TABLE I.
SUMMARY OF DATA SETS.

name	instances	attributes	minority
cyyoung8002	189	7	23%
sonar	208	60	47%
biomed	209	8	36%
bodyfat-bin	252	13	19%
vote	435	16	39%
credit	490	15	44%
boston	506	13	26%
hprice	546	11	50%
breast-w	699	9	34%
diabetes	768	8	35%

We extended WEKA [25] in order to implement our ensemble construction procedure described earlier. For each ensemble, the number of member classifiers was set to 10, and all of them were trained by the C4.5 decision tree algorithm [26]. Ensembles of decision trees are commonly used [27], [28], [29].

In Table II, we report the results from 5x2-f CV (5 iterations of 2-fold cross-validation), which is commonly used by machine learning researchers and practitioners especially when the data sets used in performance evaluation are small. In the first row of header, Accuracy and F1-measure indicate results measured in accuracy and F1-measure, respectively; in the second row of the header, S and NS refer to using sampled and unsampled data instances to evaluate member classifiers in weight assignment, respectively; in the second column, Acc and F1 refer to using accuracy and F1-measure in weight assignment, respectively; in the third column, B is for bagging (baseline), while L and E are for the logarithmic and exponential functions, respectively. For a data set, we report identical results given by bagging, to which S and NS are not applicable, for better presentation. In each cell, the number is obtained by averaging results from 5 iterations. In each column, the best results for a data set are in bold text.

TABLE II.
5X2-F CV RESULTS IN ACCURACY AND F1-MEASURE.

			Accuracy		F1-measure	
			S	NS	S	NS
cyyoung8002		B	.844	.844	.598	.598
	Acc	L	.852	.853	.64	.641
		E	.851	**.854**	.638	**.645**
	F1	L	**.853**	.853	**.646**	.644
		E	**.853**	.853	**.646**	.644
sonar		B	**.745**	.745	**.73**	**.73**
	Acc	L	.742	.742	.709	.714
		E	.743	.742	.709	.714
	F1	L	.738	**.746**	.705	.717
		E	.738	**.746**	.704	.717
biomed		B	.881	.881	.834	**.834**
	Acc	L	.886	**.884**	.836	.833
		E	**.887**	**.884**	**.837**	.833
	F1	L	.884	**.884**	.834	.833
		E	.884	**.884**	.834	.833
bodyfat-bin		B	**.871**	**.871**	.608	.608
	Acc	L	.869	.863	**.621**	.621
		E	.869	.864	**.621**	**.625**
	F1	L	.868	.869	**.621**	**.625**
		E	.868	.869	**.621**	.622
vote		B	**.96**	**.96**	**.949**	**.949**
	Acc	L	**.96**	**.96**	.948	**.949**
		E	**.96**	**.96**	.948	**.949**
	F1	L	.959	**.96**	.948	**.949**
		E	.959	**.96**	.948	**.949**
credit		B	.867	.867	**.852**	.852
	Acc	L	**.869**	**.871**	**.852**	.855
		E	**.869**	**.871**	**.852**	.855
	F1	L	**.869**	**.871**	**.852**	**.856**
		E	**.869**	**.871**	**.852**	.855
boston		B	**.912**	**.912**	.823	.823
	Acc	L	.911	**.912**	**.825**	.827
		E	.911	**.912**	**.825**	.827
	F1	L	.91	.911	.824	.826
		E	.91	**.912**	.824	**.828**
hprice		B	.782	.782	.767	.767
	Acc	L	**.787**	**.786**	**.777**	.776
		E	**.787**	**.786**	**.777**	.776
	F1	L	.786	**.786**	.776	**.777**
		E	.786	**.786**	.776	**.777**
breast-w		B	.957	.957	.937	.937
	Acc	L	**.958**	**.958**	**.939**	.939
		E	**.958**	**.958**	**.939**	.939
	F1	L	**.958**	**.958**	**.939**	**.94**

diabetes		E	**.958**	**.958**	**.939**	.939
	Acc	B	.749	.749	.606	.606
		L	**.754**	.75	**.631**	.625
		E	**.754**	.751	**.631**	.626
	F1	L	.752	**.752**	**.631**	**.631**
		E	.752	.751	**.631**	.63

From Table II, we have the following findings: First, we can observe minor improvements over bagging, while sometimes a small increase in accuracy could be of a great value in practice. Second, for weight assignment, using accuracy is slightly better than using F1-measure, and using unsampled data instances is slightly better than using sampled ones. Third, using the exponential function is slightly better than using the logarithmic function.

Furthermore, we report the results from 10x10-f CV (10 iterations of 10-fold cross-validation) in Table III.

TABLE III.
10X10-F CV RESULTS IN ACCURACY AND F1-MEASURE.

			Accuracy		F1-measure	
			S	NS	S	NS
cyyoung8002	Acc	B	0.851	0.851	0.63	0.63
		L	**0.856**	**0.858**	**0.66**	**0.666**
		E	**0.856**	**0.858**	**0.66**	0.664
	F1	L	**0.856**	0.856	**0.66**	0.662
		E	**0.856**	0.857	**0.66**	0.663
sonar	Acc	B	0.775	0.775	**0.762**	**0.762**
		L	0.775	**0.777**	0.748	0.75
		E	0.775	**0.777**	0.748	0.75
	F1	L	**0.776**	0.776	0.749	0.749
		E	**0.776**	0.776	0.749	0.75
biomed	Acc	B	0.907	0.907	0.864	0.864
		L	0.908	**0.91**	0.865	**0.868**
		E	0.908	**0.91**	0.865	**0.868**
	F1	L	**0.909**	0.909	**0.866**	0.867
		E	**0.909**	**0.91**	**0.866**	**0.868**
bodyfat-bin	Acc	B	**0.866**	**0.866**	0.593	0.593
		L	0.864	0.863	0.604	0.601
		E	0.863	0.864	0.603	0.603
	F1	L	**0.866**	0.863	**0.612**	**0.605**
		E	**0.866**	0.863	**0.612**	**0.605**
vote	Acc	B	0.961	0.961	0.95	0.95
		L	**0.963**	**0.963**	**0.952**	**0.952**
		E	**0.963**	**0.963**	**0.952**	**0.952**
	F1	L	**0.963**	**0.963**	**0.952**	**0.952**
		E	**0.963**	**0.963**	**0.952**	**0.952**
credit	Acc	B	**0.879**	**0.879**	**0.865**	**0.865**
		L	0.878	0.878	0.863	0.863
		E	0.878	0.878	0.863	0.863
	F1	L	0.878	**0.879**	0.862	0.863
		E	0.878	**0.879**	0.862	0.864
boston	Acc	B	**0.919**	**0.919**	0.837	0.837
		L	**0.919**	**0.919**	0.839	**0.84**
		E	**0.919**	**0.919**	**0.84**	**0.84**
	F1	L	**0.919**	**0.919**	**0.84**	**0.84**
		E	**0.919**	0.918	**0.84**	0.839
hprice	Acc	B	**0.789**	**0.789**	0.774	0.774
		L	0.787	0.786	0.777	0.776
		E	0.786	0.786	0.777	0.776
	F1	L	0.787	0.786	0.777	**0.777**
		E	0.787	0.786	**0.778**	**0.777**
breast-w	Acc	B	0.959	0.959	0.941	0.941
		L	**0.961**	**0.961**	**0.943**	**0.944**
	F1	E	**0.961**	**0.961**	**0.943**	**0.944**
		L	0.96	**0.961**	**0.943**	**0.944**
		E	0.96	**0.961**	**0.943**	**0.944**
diabetes	Acc	B	**0.755**	**0.755**	0.616	0.616
		L	0.752	0.754	0.63	0.633
		E	0.751	0.753	0.629	0.633
	F1	L	0.751	0.753	**0.632**	**0.635**
		E	0.751	0.753	0.631	0.634

Likewise, 10-fold cross-validation is also a common approach used to evaluate the performance of a classifier. Compared to 2-fold cross-validation, 10-fold cross-validation is used to evaluate the performance of a classifier under the situation where the portion of a given data set used for training is much larger than the portion of the data set used for testing.

The format of Table III is the same as that of Table II, and the notations used in Table III are the same as those used in Table II. Similarly, in each column of Table III, the best results for a data set are in bold text; however, the number in each cell is obtained by averaging results from 10 iterations.

From Table III, we can observe minor improvements given by the presented weight assignment and adjustment procedures over bagging. For example, improvements over bagging can be observed in *cyyoung8002*, *biomed*, *vote*, and *breast-w*, no matter accuracy or F1-measure is used in evaluation, no matter if unsampled data instances are used in weight assignment, and no matter what function is used in weight adjustment. For weight assignment, when considering the same data set and the same weight adjustment function, we find that the final results given by using F1-measure are slightly better than those given by using accuracy. We also find that using unsampled data instances in weight assignment is slightly better than using sampled ones. Finally, we find it difficult to conclude if using the exponential function in weight adjustment is better than using the logarithmic function.

V. CONCLUSIONS AND FUTURE WORK

For classification, an ensemble consists of multiple member classifiers, and it generates an overall outcome by aggregating outcomes generated by member classifiers for a testing data instance. A common approach to aggregation is to use voting. It usually shows good performance and has been applied in various applications. A common extension of the simple voting process is the weighted voting process. However, most related studies focus more on weight assignment rather than weight adjustment. In this paper, we present our study on weight adjustment functions. For each member classifier of an ensemble, a weight is initially assigned to its votes according to the performance in accuracy or F1-measure that it achieved in training; the weights of member classifiers are then adjusted according to the relative performance of each member classifier and the relative difficulty of each data set used in training. This paper is different from others in that each of the studied functions

determines nonlinearly both the performance of a member classifier and the difficulty of a data set.

The future work of this paper is as follows: First, we would like to study the use of measures other than accuracy and F1-measure in weight assignment. Second, we would like to study weight adjustment functions different from those studied in this paper. Moreover, a recently proposed technique takes into account the characteristic of the training data set and the characteristic of the underlying algorithm when constructing an ensemble [30]. It would be worth investigating the integration of the weight assignment and adjustment procedures presented in this paper into such a technique, in order to construct an ensemble better suitable for the given data sets.

ACKNOWLEDGMENT

The work presented in this paper was supported in part by the National Science Council of Taiwan under Grant Number NSC 101-2221-E-004-011. This paper was also supported in part by "Aim for the Top University Plan" of the National Chengchi University, Taipei, Taiwan, and the Ministry of Education of Taiwan. Their support is gratefully acknowledged. The author would also like to thank anonymous reviewers for their precious time.

REFERENCES

[1] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169-198, 1999.

[2] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Proceedings of International Workshop on Multiple Classifier Systems*, pp. 1-15, 2000.

[3] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1-39, 2010.

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

[5] P. Buhlmann and B. Yu, "Analyzing Bagging," *The Annals of Statistics*, vol. 30, no. 4, pp. 927-961, 2002.

[6] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning*, vol. 36, no. 1-2, pp. 105-139, 1999.

[7] A. F. R. Rahman, H. Alam, and M. C. Fairhurst, "Multiple Classifier Combination for Character Recognition: Revisiting the Majority Voting System and Its Variations," in *Proceedings of International Workshop on Document Analysis Systems*, Springer-Verlag, London, UK, pp. 167-178, 2002.

[8] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *Proceedings of International Conference on Multiple Classifier Systems*, Springer-Verlag, Berlin, Heidelberg, pp. 350-359, 2011.

[9] G. Tsoumakas, I. Partalas, and I. Vlahavas, "A taxonomy and short review of ensemble selection," in *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, 2008.

[10] N. C. Oza and K. Tumer, "Classifier ensembles: Select real-world applications," *Information Fusion*, vol. 9, no. 1, pp. 4-20, 2008.

[11] L. I. Kuncheva and J. J. Rodriguez," A weighted voting framework for classifiers ensembles," *Knowledge and Information Systems*, pp. 1-17, 2012.

[12] H. Kim, H. Kim, H. Moon, and H. Ahn," A weight-adjusted voting algorithm for ensembles of classifiers," *Journal of the Korean Statistical Society*, vol. 40, pp. 437-449, 2011.

[13] N. Stepenosky, J. Kounios, C. Clark, and R. Polikar, "Ensemble Techniques with Weighted Combination Rules for Early Diagnosis of Alzheimer's Disease," in *Proceedings of International Joint Conference on Neural Networks*, pp. 1935-1942, 2006.

[14] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana, "On the effect of calibration in classifier combination," *Applied Intelligence*, vol. 38, no. 4, pp. 566-585, 2013.

[15] N. Littlestone and M. K. Warmuth, "The Weighted Majority Algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212-261, 1994.

[16] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective Voting of Heterogeneous Classifiers," in *Proceedings of European Conference on Machine Learning*, pp. 465-476, 2004.

[17] F. Moreno-Seco, J. M. Iñesta, P. J. P. de León, and L. Micó, "Comparison of classifier fusion methods for classification in pattern recognition tasks," in *Proceedings of Joint IAPR International Conference on Structural, Syntactic, and Statistical Pattern Recognition*, pp. 705-713, 2006.

[18] L.-M. He, X.-B. Yang, and F.-S. Kong, "Support Vector Machines Ensemble with Optimizing Weights by Genetic Algorithm," in *Proceedings of International Conference on Machine Learning and Cybernetics*, pp. 3503-3507, 2006.

[19] Z.-Q. Shen and F.-S. Kong, "Optimizing Weights by Genetic Algorithm for Neural Network Ensemble," in *Proceedings of International Symposium on Neural Networks*, pp. 323-331, 2004.

[20] R. Chen and J. Yu, "An improved bagging neural network ensemble algorithm and its application," in *International Conference on Natural Computation*, 2007.

[21] R. Polikar, S. Krause, and L. Burd L, "Dynamic weight update in weighted majority voting for Learn++," in *Proceedings of International Joint Conference on Neural Networks*, pp. 2770-2775, 2003.

[22] A. Gangardiwala and R. Polikar, "Dynamically weighted majority voting for incremental learning and comparison of three boosting based approaches," in *Proceedings of International Joint Conference on Neural Networks*, pp. 1131-1136, 2005.

[23] K. Bache and M. Lichman. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. 2013.

[24] P. Vlachos. StatLib datasets archive. [http://lib.stat.cmu.edu/datasets]. 2005.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten," The WEKA Data Mining Software: An Update," *ACM SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009.

[26] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[27] T. G. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Machine Learning*, vol. 40, no. 2, pp. 139-157, 2000.

[28] K. Machova, F. Barcak, and P. Bednar, "A Bagging Method using Decision Trees in the Role of Base Classifiers," *Acta Polytechnica Hungarica*, vol. 3, no. 2, pp. 121-132, 2006.

[29] S. Pramanik, U. N. Chowdhury, B. K. Pramanik, and N. Huda, "A Comparative Study of Bagging, Boosting, and C4.5: The Recent Improvements in Decision Tree Learning Algorithm," *Asian Journal of Information Technology*, vol. 9, no. 6, pp. 300-306, 2010.

[30] K.-W. Hsu, "Weight-adjusted bagging of classification algorithms sensitive to missing values," *International Journal of Information and Education Technology*, vol. 3, no. 5, pp. 560-566, 2013.