

Exploiting Diversity in Ensembles: Improving the Performance on Unbalanced Datasets

Nitesh V. Chawla and Jared Sylvester

Department of Computer Science and Engineering
University of Notre Dame, IN 46556, USA
{nchawla, jsylvest}@nd.edu

Abstract. Ensembles are often capable of greater predictive performance than any of their individual classifiers. Despite the need for classifiers to make different kinds of errors, the majority voting scheme, typically used, treats each classifier as though it contributed equally to the group’s performance. This can be particularly limiting on unbalanced datasets, as one is more interested in complementing classifiers that can assist in improving the true positive rate without significantly increasing the false positive rate. Therefore, we implement a genetic algorithm based framework to weight the contribution of each classifier by an appropriate fitness function, such that the classifiers that complement each other on the unbalanced dataset are preferred, resulting in significantly improved performances. The proposed framework can be built on top of any collection of classifiers with different fitness functions.

1 Introduction

Ensemble systems are becoming increasingly important as they have repeatedly demonstrated the capacity to improve upon the accuracy of a single classifier in practice and theory [1–3]. Ensembles can either be homogeneous, in which every base classifier is constructed with the same algorithm, or heterogeneous, in which different algorithms are used to learn the ensemble members. Once multiple classifiers have been constructed, their predictions are typically combined through some method of voting. Many of the popular ensemble techniques use a simple majority vote of every classifier. It stands to reason that not every member contributes equally to the group performance, and as such, not every member should be given equal weight in the decision making. However, the question remains how best to determine the relative worth of every classifier.

A simple solution would be to weight each individual according to its accuracy on some validation set. However, for an ensemble to be more accurate than its members, they need only be more accurate than a random guess. It is not necessarily true that the most accurate classifiers contribute the most to an ensemble — diversity also plays an important role [4, 3]. Diversity is the degree to which classifiers disagree in the errors they make. This allows the voted accuracy to be greater than the accuracy of any single classifier.

Learning from unbalanced datasets can impose more exacting requirements on the diversity in types of errors among classifiers, because we are interested in the collection of classifiers that help in reducing the false negatives without significantly increasing the false positives. There is a non-uniformity in the types of errors that can be made, given the interest in increasing the true positives even if at the expense of increasing the false positives. Moreover, the classifiers themselves are learned with the objective criterion of reducing the overall error, and not necessarily the error on the positive class, which is typically the minority class. The goal then is to be able to discover a weighting scheme for individual classifiers such that the collective classification helps in improving the true positive rate. One could potentially weight each classifier by a metric that is more amenable to evaluating classifiers on unbalanced datasets, but that does not help in realization of diversity. In this case, each classifier is independently evaluated on a validation set and that independent evaluation becomes the weight, but it does not help in realization of their performance as a collective.

Thus, the problem we face is: *how to best set the weights of each classifier to maximize the performance of the entire ensemble on unbalanced datasets?* To that effect, we utilize an evolutionary framework using a genetic algorithm to assign weights for each classifier in an ensemble. The goal is to assign weights that reflect the relative contribution of each classifier in improving the overall performance of the ensemble. The genetic algorithm starts with random weight assignment to the classifiers, and after multiple generations, these weights come to reflect the relative contribution of the corresponding classifiers. Such an evolutionary combination of members can be more effective than considering all members equally. We generated the weights using the positive (or minority) class *f-measure* [5] as the fitness score. It is more attuned with the evaluation on unbalanced datasets. Furthermore, by separating the stages of learning classifiers and forming ensemble, our meta-learner does not need any knowledge of the structure or type of the base classifiers, just their predictions.

Contributions: We posited the following questions in our evaluation of ensembles learned on unbalanced datasets.

- What is the improvement offered by the proposed evolutionary framework over uniform voting, weighted voting, and the best member-classifier of the ensemble?
- What is the impact of using a technique designed for countering imbalance in data as a pre-processing stage before learning ensembles? We used an over-sampling method, SMOTE [6], that generates synthetic minority (positive) class instances.
- Is there a relationship among the classifiers selected by our proposed framework and the classifiers on the ROC convex hull [7]?

1.1 Related Work

Genetic algorithms have a history of use in data mining, especially as wrapper technique of feature selections [8–10] that use genetic algorithms as an iterative

search technique to identify the best performing feature subset on a validation set. Kim et al. [11] proposed creating meta-evolutionary ensembles, in which both classifiers and ensembles are evolved simultaneously. However, only neural networks were considered as potential classifiers. In a similar vein, Menczer et al. [12] proposed a local selection methodology for evolving neural networks in which they seek to minimize the interaction among members. Liu et al. [13] also explored evolutionary techniques for neural networks. They evolved both ensembles and neural networks simultaneously, and focused on speciation to minimize the interactions between ensemble members. Kuncheva and Jain [14] designed a multiple classifier fusion system using genetic algorithms, albeit genetic algorithms were used to first select the features and then on those selected feature subsets they learned three different types of classifiers (linear and quadratic discriminant classifiers and the logistic classifier). None of these methods focused on learning from unbalanced datasets. Moreover, the variety of classifiers considered in the ensemble was very limited.

2 Building Evolutionary Ensembles

We implemented a framework that we call EVEN (EVolutionary ENsembles) [15], which uses a genetic algorithm to carry out the meta-learning. EVEN utilizes the individual classifier predictions on the validation set as input. EVEN's output comprises weights for each individual classifier and the final composite testing set predictions based on those weights.

Algorithm *Evolution*(G, p, C)

Input: Number of generations G ; Population size p ; Number of classifiers C

Output: Weight vector W ; Predictions X

1. (* Let P_g denote the population in generation g , and let $P_g.i$ be *member* _{i} of that population. *)
2. $P_0 = \text{Random_population}(p)$
3. **for** $g \leftarrow 1$ **to** G
4. **for** $i \leftarrow 0$ **to** p
5. Compute: $\text{fitness}_i = \text{fmeasure}(P_g.i)$
6. **endfor**
7. Sort P_g by *fitness*
8. $P_{g+1} = \text{interbreed}(P_g) + \text{mutations}(P_g) + \text{survivors}(P_g)$
9. **endfor**
10. Select $P_{G-1}.i$ from P_{G-1} such that
 $\text{fitness}(P_{G-1}.i) = \max(\text{fitness}(P_{G-1}.1), \dots, \text{fitness}(P_{G-1}.p))$
11. **return** $W = \text{weights}(P_{G-1}.i)$
12. **return** $Y = \text{predictions}(P_{G-1}.i)$

EVEN is implemented as follows. A dataset is partitioned into three disjoint subsets, one for training, one for validation and one for testing. The base classifiers are learned on the training sets, and then evaluated on the validation data

and the testing data. Their predictions for both validation and testing sets were recorded. The predictions on the validation sets are used to train EVEN, and the predictions on the testing set are used to evaluate EVEN’s final performance. It is important to note that all EVEN knows of the base classifiers is the predictions they made; EVEN does not have any information regarding either the decision making process of the classifiers or the features of the original data sets. Once the appropriate weights have been learned on the validation set those weights are used to combine the predictions of the base classifiers made on the testing set. The weight vector of classifiers as generated by EVEN is $W = w_1, w_2, \dots, w_C$, where C is the number of classifiers in the ensemble. Then the prediction for a test instance j :

$$\hat{y}_j = \sum_{i=1}^C w_i \times y_{i,j} \quad (1)$$

where $y_{i,j}$ is the prediction by classifier i on the test instance j . The pseudo-code for EVEN is contained in *Evolution*.

3 Experiments

We ran a variety of experiments to demonstrate the efficacy of EVEN. We used heterogeneous ensembles comprised of different learning techniques as individual members, and homogeneous ensembles comprising the same underlying learning technique but with each classifier trained on a modified version of the dataset. We included a variety of datasets with different sizes and characteristics. We preprocessed the datasets by randomly dividing each into the training/validation/testing sets *ten* different times, and ran as many experiments.

3.1 Datasets

Our experiments featured five publicly available datasets stemming from different domains with differing levels of class imbalance. Table 1 shows the varying characteristics of the datasets, which are comprised of a mixture of continuous and nominal values.

Table 1. Datasets. Ordered by the increasing amount of skew.

Dataset	Number of examples	Number of Features	Class Distribution (Negative:Positive)
Pima	768	8	0.6510:0.3490
Phoneme	5404	5	0.7065:0.2935
Satimage	6435	36	0.9027:0.0973
Covtype	35,754	54	0.9287:0.0713
Mammography	11,183	6	0.9768:0.0232

3.2 Experiment-1: Heterogeneous Ensemble On Unbalanced Datasets

We used 12 different learning algorithms, as implemented in Weka, for constructing the base ensemble: ID3 decision trees, J48 decision trees (C4.5), JRIP rule learner (Ripper), Naïve Bayes, NBTree (Naïve Bayes trees), 1R, logistic model trees, logistic regression, decision stumps, multi-layer perceptron, SMO (support vector machine), and 1BK (k-nearest neighbor). We chose to work with many different classification algorithms because each displays a different inductive bias, and therefore provides a potentially more independent and diverse set of predictions to build upon. In order to enlarge the ensembles to more interesting sizes, ten full size bootstraps were made on the training data. Each classifier was learned on each bootstrap, creating a total of 120 classifiers for EVEN to work with. The purpose of constructing bootstraps was just to generate more members of the ensembles. For each of the runs, EVEN maintained a population of size 250 for 1000 generations.

Our experimental set-up included results from EVEN voted classifiers to an un-weighted uniform vote of classifier; a weighted vote where the weight was equal to the validation set accuracy of the classifier; a weighted vote where the weight was set equal to the validation set measure of each classifier; and the *f-measure* of the best performing single classifier. We chose to compare to a simple majority vote because that is the most common voting form in ensembles. The weighted vote was included to verify that the classifiers' individual performance was not a sufficient indicator of its relative contribution. Finally, the single best classifier, as defined by performance on the validation set, was also used for comparison to ensure that the added overhead of using multiple classifiers was justified.

However, for space considerations, we elided the results on the accuracy weighted vote, as the *f-measure* and uniformly voted classifiers sufficed for the comparisons with EVEN. *f-measure* weighted voting outperformed accuracy weighted voting, as one would expect, and there was no statistical difference, if any, between uniform voting and accuracy weighted voting. However, we do report all the statistical significance comparisons, subsequently. Table 2 shows the average *f-measure* and the standard deviations. It is evident from Table 2 that EVEN obtains the best average *f-measure* as compared to the other considered schemes.

Table 2. *f-measure* of ensemble of 120 classifiers. The following convention is used in the Table: FMW-V is *f-measure* weighted voting; U-V is uniform voting; and Best is the best classifier in the ensemble.

Dataset	EVEN	FMW-V	U-V	Best
Pima	0.6528 ± 0.0705	0.6411 ± 0.0561	0.6315 ± 0.0546	0.6345 ± 0.0600
Phoneme	0.8221 ± 0.0373	0.7553 ± 0.0182	0.7351 ± 0.0174	0.7849 ± 0.0435
Satimage	0.6682 ± 0.0369	0.6636 ± 0.0402	0.4625 ± 0.0525	0.6599 ± 0.0328
Covtype	0.9022 ± 0.0157	0.8828 ± 0.0086	0.8508 ± 0.0120	0.8748 ± 0.0080
Mammography	0.6186 ± 0.0590	0.5950 ± 0.0501	0.5669 ± 0.0529	0.5731 ± 0.0750

EVEN always significantly outperforms uniform voting and accuracy weighted voting. It statistically significantly better than *f-measure* weighted voting in 4 out of 5 datasets. This presents empirical evidence that it is not the individual worth of a classifier that provides the classification improvement, but it is their relative worth as a collective. As an example, we show the weights versus *f-measure* distribution for the phoneme dataset in Figure 1. Firstly, there is no obvious correlation observed between the weights and the corresponding *f-measure*. Classifiers with similar *f-measures* have a range of weights, and the classifiers with higher weights can also have lower *f-measures*. Similar trends were observed for the other datasets.

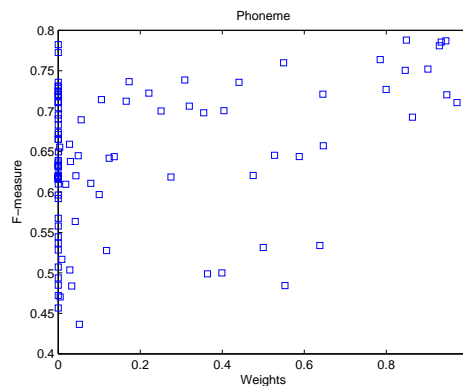


Fig. 1. Weights versus *f-measure* distribution across all the classifiers for the Phoneme dataset.

We are more interested in classifiers that make different types of errors, canceling out each others mistakes, leading to an overall improvement in the ensemble’s performance. Based on these weights one can also establish a cut-off for the classifiers that will vote together, thus reducing the size of the ensemble, which can be very relevant when the ensemble sizes are in thousands. To evaluate this, we established a cut-off value of 0.8 on the weights. Table 3 shows the number of the selected classifiers and the corresponding average *f-measure*. Applying the cut-off results in a significantly reduced number of classifiers, approximately 20 – 25% of the original ensemble size, for all the datasets, without any significant drop in the *f-measure*. This shows the effectiveness of EVEN in assigning relevant weights to the classifiers and identifying the maximally contributing subset.

In accordance with the Free Lunch Theorem [16], no single classifier was consistently best on all the datasets or even within the set of runs within a single dataset. Thus, it becomes very important to have a collection of classifiers to apply to any given dataset. But if one utilizes an ensemble voting framework like ours, then the selection of a single classifier becomes unimportant, as each classifier is assigned a weight based on its relative contribution in the collective.

Table 3. Average *f-measure* and Size (including standard deviations) of ensemble after applying a cutoff value of 0.8 to classifier weights in the original ensemble of 120 classifiers.

Dataset	EVEN _{0.8}	# Classifiers
Pima	0.6483 ± 0.0650	27.9 ± 5.7436
Phoneme	0.8142 ± 0.0194	24.5 ± 3.8370
Satimage	0.6696 ± 0.0430	28.4 ± 2.8752
Covtype	0.9079 ± 0.0122	27.3 ± 4.2177
Mammography	0.5817 ± 0.0921	27.8 ± 4.4096

Table 4. *f-measure* of ensemble of classifiers learned after applying SMOTE. Same convention as Table 2 applies.

Dataset	EVEN	FMW-V	U-V	Best
Pima	0.7027 ± 0.0515	0.6966 ± 0.0550	0.7002 ± 0.052	0.6874 ± 0.0569
Phoneme	0.8160 ± 0.0161	0.7742 ± 0.0147	0.7653 ± 0.0137	0.8193 ± 0.0223
Satimage	0.6459 ± 0.0403	0.6663 ± 0.0443	0.5816 ± 0.0421	0.6560 ± 0.0490
Covtype	0.9063 ± 0.0202	0.8954 ± 0.0135	0.8611 ± 0.0189	0.8865 ± 0.0261
Mammography	0.6937 ± 0.1122	0.6655 ± 0.1194	0.6649 ± 0.1223	0.6172 ± 0.1217

3.3 Experiment-2: Effect Of SMOTE On Ensemble Generation

For this set of experiments we did not bootstrap the data. Instead we applied SMOTE at 5 different levels, 100%, 200%, 300%, 400%, and 500%, to synthetically generate new positive class instances. This resulted in different dataset sizes and class distributions for members of the ensemble. We also included classifiers learned on the original distribution, thus giving us a total of 6 classifiers on a dataset. This resulted in an ensemble of 72 classifiers. The main purpose of this experiment was to study the behavior of the ensemble when the dataset has been pre-treated with a technique to counter the class imbalance and its impact on different voting schemes. We used exactly the same training, validation, and testing sets as in Experiment-1. This allowed easy juxtaposing of the two results from both the experiments. Table 4 shows the results on this set of experiments. Again, we included the same set of comparisons as in the previous subsection. It is evident that the initial preprocessing of data by applying an oversampling technique such as SMOTE, clearly benefits all the voting schemes. Table 5 shows the table of statistical significance comparisons. The surface of statistical significance now changes, as all the classifiers are able to capitalize on the oversampling provided by SMOTE.

The most significant improvements are for uniform voting and accuracy weighted voting. This is not surprising as now the classifier’s inductive bias is manipulated towards the minority class because of oversampling, which leads to improved true positives, but at the expense of false positives. We also note that there is not much difference in the *f-measure* values obtained by EVEN in Experiment-1 and Experiment-2. This is very interesting in the light of improve-

ments offered by SMOTE in all other schemes. The best classifier performance also improves. EVEN is, nevertheless, able to capture the diversity among the classifiers in their predictions and exploit it, despite classifiers’ inductive bias towards the majority class. If we compare the results on Experiment-2 and Experiment-1 (as the testing sets remain the same), SMOTE with EVEN helps in performance improvement only for two datasets — pima and mammography. EVEN is thus able to exploit the differences in the errors to generate weights for optimizing the *f-measure* performance, irrespective of the prior oversampling.

Table 5. Table of statistical significance.

Dataset	AW-V	FMW-V	U-V	Best
Pima	–	–	–	–
Phoneme	Y	Y	Y	–
Satimage	Y	–	Y	–
Covtype	Y	Y	–	Y
Mammography	–	–	–	Y

3.4 EVEN And ROC Space

We are also interested in understanding the classifiers selected via EVEN for unbalanced datasets and their relationship in the ROC space. Ideally, the ROC convex hull represents the family of classifiers that are optimal in the different operating conditions [7]. The compelling question then is: *How many of the higher weighted EVEN classifiers lie on the ROC convex hull?* To that end, we use C4.5 decision trees from our family of classifiers to understand the phenomenon. We smoothed the C4.5 leaf frequencies using the LaPlace estimate [17], and then constructed ROC curves from the leaf estimates. Moreover, having just a single type of classifier in the ensemble made the ROC space more tractable for the purpose of this experiment. Now to construct an ensemble, we first applied different levels of SMOTE to each of the datasets.

As an example, Figure 2 shows the family of ROC curves and the ROC convex hull for mammography dataset. Due to space considerations, we only show the ROC curves for the most unbalanced dataset. Clearly, not all classifiers are falling on the ROC convex hull. *SMOTE* = 500 classifier is the most dominating one with the most number of points on the ROC space (approximately 72% of the ROC convex hull is occupied by points belonging to *SMOTE* = 500). That particular classifier also carried the maximum voting weight assigned by EVEN: 0.99521. But, one might ask what if we just vote the classifiers on the convex hull. The *f-measure* then obtained is 0.56, whereas the *f-measure* obtained by EVEN is 0.592 (a 5.7% improvement offered by EVEN). It is indicative of the value added by the EVEN weighted participation of all the classifiers.

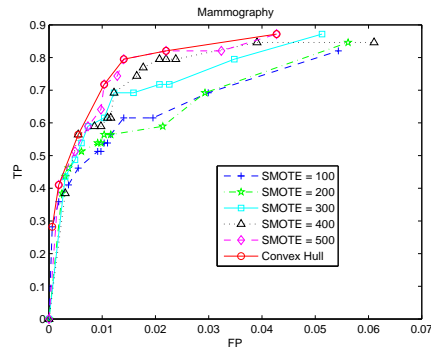


Fig. 2. ROC curves for Mammography dataset.

4 Conclusions

Ensemble generation using EVEN offers a significant advantage over the different voting schemes and the best classifier member of the ensemble. EVEN is effectively able to weight each individual classifier of the ensemble on the basis of its relative contribution to the overall performance, and these weights carry no obvious correlation to the independent performance of the classifier. This underlines the fact that the key is not the individual merit of the classifier, but their diversity in making different types of errors. We also generated ensembles after applying SMOTE, which significantly helped all the voting methods. Essentially, we are introducing inherent diversity in the SMOTE-based ensembles by learning classifiers on different amounts and instances of SMOTE. This increases the capability of the ensemble to optimize on f -measure, since the individual classifiers are learned on different class distributions and demonstrate different biases in their predictions.

As part of future work, we are also incorporating other metrics such as AU-ROC [18] and g-mean within the classifier selection framework. A framework like EVEN allows for incorporation of any evaluation metric, given the data/class distribution. This offers flexibility to incorporate any evaluation metric as a fitness function, since the classifiers don't have to be re-learned; all we are interested in is the relative contribution of the classifier given a different objective function.

References

1. Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Thirteenth International Conference on Machine Learning*, 1996.
2. L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

3. T. Dietterich, "Ensemble methods in machine learning," *Lecture Notes in Computer Science*, vol. 1857, pp. 1–15, 2000.
4. L. Kuncheva and C. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, pp. 181–207, 2003.
5. C. J. van Rijsbergen, *Information Retrieval*. London: Butterworths, 1979.
6. N. Chawla, L. Hall, B. K.W., and W. Kegelmeyer, "SMOTE: Synthetic Minority Oversampling TEchnique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
7. F. Provost and T. Fawcett, "Robust Classification for Imprecise Environments," *Machine Learning*, vol. 42/3, pp. 203–231, 2001.
8. D. Opitz, "Feature selection for ensembles," in *AAAI/IAAI*, pp. 379–384, 1999.
9. C. Guerra-Salcedo and L. Whitley, "Genetic approach to feature selection for ensemble creation," in *International Conference on Genetic and Evolutionary Computation*, pp. 236–243, 1999.
10. J. Yang and V. Honavar, "Feature subset selection using A genetic algorithm," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*, p. 380, 13–16 1997.
11. Y. Kim, N. Street, and F. Menczer, "Meta-evolutionary ensembles," in *IEEE Intl. Joint Conf. on Neural Networks*, pp. 2791–2796, 2002.
12. F. Menczer, W. N. Street, and M. Degeratu, "Evolving heterogeneous neural agents by local selection," in *Advances in the Evolutionary Synthesis of Neural Systems* (V. Honavar, M. Patel, and K. Balakrishnan, eds.), Cambridge, MA: MIT Press, 2000.
13. Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEE Transactions on Evolutionary Computation*, vol. 4.4, pp. 380–387, 2000.
14. L. I. Kuncheva and L. C. Jain, "Designing classifier fusion systems by genetic algorithms," *IEEE-EC*, vol. 4, pp. 327 – 336, November 2000.
15. J. Sylvester and N. V. Chawla, "Evolutionary ensemble creation and thinning," in *International Joint Conference on Neural Networks*, pp. 5148 – 5155, 2006.
16. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, April 1997.
17. F. Provost and P. Domingos, "Tree induction for probability-based rankings," *Machine Learning*, vol. 52(3), 2003.
18. W. B. Langdon and B. F. Buxton, "Genetic programming for combining classifiers," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 66–73, 2001.