

Evolutionary Ensemble Creation and Thinning

Jared Sylvester and Nitesh V. Chawla, *Member, IEEE*

Abstract—Ensembles are often capable of greater predictive accuracy than any of their individual members. One key attribute of ensembles’ success is the notion of diversity. However, the majority voting scheme used in most ensembles treats each classifier as if it contributed equally to the group performance, without capitalizing on the relative improvement offered by each member of the ensemble. Our solution to this problem is to use genetic algorithms to weight the contribution of each classifier. This improves the performance of the ensemble by providing a weighted vote which maximizes collaboration among classifiers. Our approach provides a general-purpose framework for evolutionary ensembles, allowing them to build on top of any collection of classifiers, whether they be heterogeneous or homogeneous. In addition, the ability of our framework to thin ensembles, and its effect on ensemble diversity is presented.

I. INTRODUCTION

Ensemble systems are becoming increasingly important in the data mining community as they have repeatedly demonstrated the capacity to improve upon the accuracy of a single classifier in practice and theory [1], [2], [3], [4], [5], [6], [7]. Popular ensemble creation techniques include bagging, random forests, random trees, random subspaces and boosting. Ensembles may be either 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 most popular ensemble techniques use a simple majority vote with uniform weight attached to each 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. There is not necessarily a correlation between using the most accurate classifiers available and an improvement in ensemble accuracy – diversity also plays an important role [8], [5], [9]. Diversity is the degree to which classifiers disagree in the errors they make. High diversity allows the voted accuracy to be greater than the accuracy of any single classifier. Because certain classifiers may be less accurate, but make different kinds of errors, assigning weights strictly according to accuracy can overlook an important factor in the success of an ensemble [10]. The compelling question remains: *How to demonstrate the reasons of success of an ensemble? Can*

we attribute it to diversity or accuracy of individual members or some combination of these? We attempt to understand the behavior of an ensemble under an evolutionary framework. In doing so, we aim for weighting classifiers based on their relative contribution to the final performance, that is, the *collaborative* power.

If classifiers should not be uniformly weighted or weighted with their individual accuracy, we must seek another method to determine their weights. Genetic algorithms [11] are well suited to multi-parameter operations, which is essentially the problem we face: how to best set the weights of each classifier to maximize the predictive accuracy of the entire ensemble. Therefore, we propose using evolutionary techniques to ascertain the appropriate weights for each members vote in the final classification. Our genetic algorithm begins with random weights assigned to each classifier. 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.

Genetic algorithms have a history of use in data mining, especially as wrapper technique for feature selection[12], [13], [14]. They use genetic algorithms as an iterative search technique to identify the best performing feature subset on a validation set. Kim et al. [15] 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. [16] present a local selection methodology for evolving neural networks in which they seek to minimize the interaction among members. Liu et al. [17] also explored evolutionary techniques for neural networks. They evolved both ensembles and neural networks simultaneously, and focused on speciation to minimize the interactions between ensembles members. Kuncheva and Jain [18] designed a multiple classifier fusion system using genetic algorithms, albeit they used the genetic algorithms 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).

A. Our Contribution

This work seeks to extend the previous applications of evolutionary data mining by applying genetic algorithms to ensemble creation regardless of the type of the base classifier. We study both heterogeneous and homogeneous evolutionary ensembles, utilizing many different types of base classifiers. Furthermore, by separating the stages of learning classifiers and forming the ensemble, our meta-learner does not need any knowledge of the structure or type of the base classifiers,

Jared Sylvester and Nitesh V. Chawla are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA (phone: 574-631-3637; fax: 574-631-9260; email: {jsylvest,nchawla}@nd.edu).

just their predictions. This added flexibility opens the door to both incremental and distributed data mining applications [19]. We evaluated our approach on a variety of datasets using accuracy as the fitness score, although, any fitness score can be implemented within EVEN. We conduct an expansive evaluation for different ensemble generation techniques and eventual thinning. We utilize a variety of learning algorithms to build the heterogeneous ensembles. The homogeneous ensembles are constructed from bagged decision tree classifiers.

In addition to constructing ensembles, we also explore the application of evolution to ensemble thinning. Thinning is the attempt to remove from an ensemble the classifiers which cause errors [10]. Because the cost of testing typically increases with the addition of more classifiers to an ensemble, thinning can result in improvements in classification efficiency. Furthermore, thinning can result in a more accurate ensemble by removing the members which tend to contribute to misclassifications. The results from thinning can effectively demonstrate that it is a collective of classifiers from an ensemble that provide the impetus for the overall improvement in the performance.

II. EVOLUTIONARY ENSEMBLE: EVEN

The method we have adopted for ensemble formation is essentially a stacking system [20]. A set of classifiers are learned, and the predictions of each classifier on a validation set become the features of a new dataset. This new dataset is used to train a meta-classifier. The meta-classifier then learns a function on the prediction of the base classifiers and makes final, composite, predictions. Stacking can reduce the risk of overfitting. We implemented a system which we call EVEN (EVolutionary ENsembles), 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 is comprised of weights for each individual classifier and the final composite testing set predictions based on those weights. We will now describe EVEN in more detail.

A. Genetic Algorithms

Genetic algorithms are computational models which draw their inspiration from the evolution of biological populations. The initial genetic model developed by Holland in 1975 has been greatly extended, but the basis remains the same [21]. Potential solutions are encoded as the chromosome of some individual. A set of these individuals are instantiated as an initial population. The individuals are evaluated through some predefined fitness function. Each succeeding generation is populated by the most fit members of the previous generation and their offspring. Offspring are created through crossover and mutation. Crossover combines the genetic information of two solutions to create new children, echoing reproduction in the natural world. Mutation alters the genes of an individual to introduce more diversity into the population, allowing more solution spaces to be searched. In this way genetic algorithms provide a way for the initially random, and poor, solutions to be iteratively improved over time.

Each potential solution is encoded in EVEN as a real-valued, fixed-length array chromosome. Though Holland's initial algorithms used binary strings, real-valued chromosomes generally outperform bit-string encodings for problems which are naturally real-valued, such as ours [22]. The array is equal in length to the number of classifiers in the ensemble, and uses four bytes for each gene. Each value in the array corresponds to the weight of a classifier's vote in determining the final classification, and is normalized between zero and one. A weight of 1.0 means that the corresponding classifier is given a maximum vote in the final classification, and a weight of 0.0 means that the classifier is effectively ignored.

EVEN uses a steady state genetic algorithm (i.e. some individuals are carried over to the next generation without modification, causing overlap between generations), as implemented in the GALib library for genetic algorithms [23]. 80% of succeeding generations are generated through crossover, 1% through mutation, and 19% are "survivors" from the previous generation. A uniform crossover operator is used for interbreeding, meaning that each gene of the child takes its value by randomly selecting which parent to copy from. Thus, uniform crossover is similar to an n-point crossover, where there are n+1 genes on the chromosome. The mutation operator is a point mutation on a Gaussian distribution, meaning that after EVEN has decided a gene in a chromosome will be mutated, and it is added with a random value selected from a Gaussian distribution to get the mutated value. A population size of 250 and 1000 generations were used throughout our experiments.

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. $P_0 = \text{Random}_{p,\text{opulation}}(p)$
3. **for** $g \leftarrow 1$ **to** G
4. **for** $i \leftarrow 0$ **to** p
5. **Compute:** $\text{fitness}_i = \text{accuracy}(P_{g,i})$
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** $Y = \text{predictions}(P_{G-1,i})$

B. Learning Framework

EVEN first partitions a dataset into three disjoint subsets, one for training, one for validation and one for testing. The base classifiers are learned on the training data, and then evaluated on the validation data and finally the testing data.

Their predictions for both validation and testing sets are 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 the base classifiers made on the testing set. If C is the total number of classifiers in the ensemble, the weight vector of classifiers as generated by EVEN is $W = w_1, w_2, \dots, w_C$. Then the prediction for a test instance j is:

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

where $y_{i,j}$ is the prediction by classifier i on the test instance j .

Two different systems were used to learn the base classifiers. For our heterogeneous ensembles, we used WEKA, an open source software package which provides Java implementations for number of different learning algorithms [24]. The homogeneous ensembles were composed of bagged decision trees learned in C4.5 [25].

To accomplish thinning in EVEN, various ‘‘cut-off’’ levels are applied to the classifiers weights. That is, only those classifiers whose weights are above a certain level are allowed to participate in decision making. Those classifiers whose weights are below the cut off are effectively culled from the ensemble, reducing its overall size. EVEN outputs the weight for every classifier, allowing the user to either use an arbitrary cutoff, or do further validation experiments to determine an optimal cut-off level to meet a desired accuracy or ensemble size. When no cut-off is applied, every classifier in the ensemble contributes its weighted prediction.

It should be noted that EVEN is able to combine classifiers learned on any set of training examples, as long as the predictions are available for all instances in the validation and testing sets. This is what gives EVEN the means to combine both heterogeneous and homogeneous classifiers. This ability also implies that EVEN can be easily applied to distributed learning of extremely large datasets. Each member classifier can be constructed completely autonomously, so EVEN can be easily parallelized or distributed, without requiring any communication or sharing of inference among the classifiers [19]. This is a very desirable property, as it leads to ease of scalability and reduced computational complexity [3].

III. EXPERIMENTAL SETUP

We ran a variety of experiments to demonstrate the effectiveness of EVEN. We used heterogeneous ensembles comprising different learning techniques as individual members, and homogeneous ensembles comprising the same underlying learning technique but a random instantiation of

the training data (such as bagging). 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. These experiments were run on each of these ten random divisions to allow us to perform significance testing.

A. Datasets

Ten publicly available datasets were used for the experiments, drawn from various domains. Most of the datasets were taken from the UCI data repository, and phoneme was taken from the ELENA project [26]. As mentioned above, each dataset was randomly divided ten different times into three disjoint subsets for use in training, validation and testing. Table I shows the varying characteristics of the datasets, which are comprised of a mixture of continuous and nominal values.

B. Heterogeneous Ensembles

We used 12 different learning algorithms, as implemented in Weka, for constructing the 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 (MLP), 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 potentially provides a more independent and diverse set of predictions to build upon.

Initial experiments using one instantiation of each algorithm were inconsequential – the size of the ensemble was too small for the meta-learning to have a noticeable effect [27]. In order to enlarge the ensembles to more interesting sizes, ten full size bootstraps were made of the training data. Each algorithm learned a classifier 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. Further experiments were done with ensembles composed of those 120 classifiers and 12 classifiers which were built on the original, un-bootstrapped training data. These 132-classifier experiments provided the opportunity to compare classifiers learned on the unaltered training data and those learned on the re-sampled data. For both the ensemble sizes, EVEN maintained a population of size 250 for 1000 generations.

For purposes of comparison we also recorded the accuracy of an un-weighted vote of classifiers, a weighted vote where the weight was equal to the validation set accuracy of the classifier, stacking using a Naïve Bayes meta-learner, stacking using a logistic regression function, and the accuracy of the single classifier which performed best on the validation set. We believe the stacking approaches using Naïve Bayes and logistic regression would be the closest, conceptually, to EVEN as both the classifiers are learning functions on top of the validation set predictions provided by the individual classifiers and trying to reduce the overall error. We chose to compare to a simple majority vote because that is the

TABLE I
DATASETS AND THEIR CHARACTERISTICS.

Dataset	Training	Validation	Testing	Classes	Features
DNA	2230	478	478	3	60
Hypo	2214	474	475	2	25
KRKP	2237	479	480	2	36
LED-24	3500	750	750	10	24
Letter	14000	3000	3000	26	16
Page	3831	821	821	5	10
Pendigit	7694	1649	1649	10	16
Phoneme	3782	811	811	2	5
Satimage	4504	965	966	7	36
Sick	2640	566	566	2	29

most common form of decision making 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.

C. Homogeneous Ensembles

The homogeneous ensembles were composed of 500 bagged decision trees built using C4.5. Bagged trees are constructed by learning each tree on 100% bootstraps of the training data [1]. After all trees were constructed, EVEN was used to assign each a weight. This weighted vote was compared to the un-weighted vote of every tree, which is the way bagged ensembles are usually formed. In addition, comparisons were made to a weighted vote based on the trees' validation accuracy.

D. Thinning Experiments

For each of the ensemble varieties, we investigated EVEN's ability to thin the full ensemble. Cut-off levels of 0.4, 0.6 and 0.8 were applied to the final weights to reduce the ensemble size. The higher the cut-off value, the fewer classifiers selected. Thus, the cut-off value of 0.0 would imply that all the (weighted) classifiers were included in the final voting. These thinned ensembles were compared to the accuracy of the full ensemble, as well as that of a randomly selected subset of the available models, which was equal in size to the subset selected by EVEN. For instance, if a particular cut-off level caused EVEN to select 53 out of 120 classifiers, 53 classifiers would be selected randomly and used to form an ensemble for comparison.

E. Diversity

We used the κ statistic as the diversity measure for the ensembles [4], [8]. κ is essentially the inter-rater agreement among classifiers [28]. Assume a dataset $Z = z_1, z_2, \dots, z_N$. For a classifier C_i , an output vector of correct classifications can be constructed such that $y_i = [y_{1,i}, \dots, y_{N,i}]^T$. Thus, $y_{j,i} = 1$, if C_i correctly classifies z_j , and 0 otherwise. L is the

number of classifiers, and $l(z_j)$ is the number of classifiers from D that correctly classify z_j . The inter-rater agreement, κ can then be defined as follows.

If \bar{p} is the average individual classification accuracy: $\bar{p} = \frac{1}{NL} \sum_{j=1}^N \sum_{i=1}^L y_{j,i}$, then

$$\kappa = 1 - \frac{\frac{1}{L} \sum_{j=1}^N l(z_j)(L - l(z_j))}{N(L - 1)\bar{p}(1 - \bar{p})} \quad (2)$$

Dietterich proposed a variant of the κ statistic [4], which we will call κ_D . κ_D , defined in Equation 5, measures the degree of similarity between two classifiers, θ_1 , while subtracting off the probability that the similarity is do to chance, θ_2 . θ_1 and θ_2 are defined in Equations 3 and 4, where S is the number of classifiers, N is the number of instances and C_{ij} is the number of instances for which *classifier_n* predicted *class_i* and *classifier_m* predicted *class_j*. Each classifier in an ensemble (or a subset of the ensemble) is combined with every other classifier to compute the mean pair-wise accuracy and κ_D value. The lower the κ_D values, the higher the disagreement amongst the classifiers.

$$\Theta_1 = \frac{\sum_{i=1}^S C_{ii}}{N} \quad (3)$$

$$\Theta_2 = \sum_{i=1}^S \left(\sum_{j=1}^S \frac{C_{ij}}{N} \sum_{j=1}^S \frac{C_{ji}}{N} \right) \quad (4)$$

$$\kappa_D = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2} \quad (5)$$

IV. RESULTS

We now present the results and comparisons of EVEN with the aforementioned approaches.

A. Heterogeneous Ensembles

A full listing of results can be seen in Table II. EVEN assigns weights to classifiers based on their propensity to cooperate and work as a collective. It is very possible that classifiers with low accuracy have higher weights than classifiers with higher accuracies. Figure 1 shows one such example; the classifiers that would pass the cut-off have a

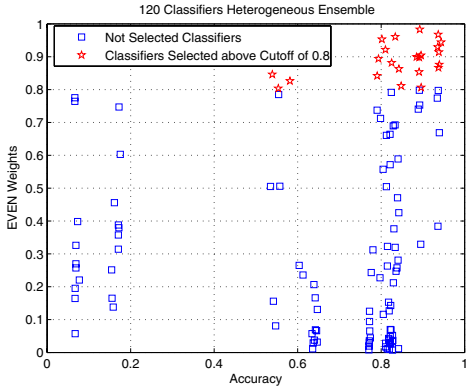


Fig. 1. Spread of Classifier Accuracy versus the EVEN assigned weights.

broad range of accuracy scores, and are not necessarily the most highly accurate ones. This is symbolic of the argument that it is the diversity among the classifiers rather than their individual aptitude that propels ensemble performance. We observed similar trends with the other datasets as well; however, due to space considerations we only include one.

Using a paired t-test at 95% confidence interval, EVEN statistically significantly outperforms majority voting for 5 out of 10 datasets (letter, pendigits, phoneme, satimage, krkp). Table III shows the number of times there was a statistically significant difference between EVEN (using all classifiers and at a cut-off of 0.8) and other approaches. If we put aside considerations of statistical significance, EVEN outperformed majority voting — both weighted and unweighted — on eight of the ten datasets and tied on one more. EVEN outperformed both of the stacking experiments as well. The results were statistically significant on 4 datasets for Naïve Bayes based stacking and on 3 datasets for logistic regression based stacking. Nevertheless, EVEN achieved higher accuracy, albeit not all were statistically significant, on 8 of the datasets.

We also evaluated the ensemble size of 132 that includes both the classifiers learned on the original dataset and the bootstraps of data. Adding the classifiers learned on the original datasets did not lead to a performance improvement. In fact, we see that the performance deteriorated slightly for some of the datasets. Interestingly, the classifiers learned on the original dataset were not selected in any additional frequencies, as compared to the classifiers learned on bootstraps. It can be attributed to their lack of sufficient contribution to reduction in the overall error with regards to the other classifiers in the set. Table IV shows the comparison between ensembles of size 120 and 132. As can be seen, the addition of classifiers learned on the original, non-bootstrapped data does not always increase the accuracy of the ensemble.

We then applied thinning to the ensembles with individual members weighted by EVEN. We applied the cut-off values $\in \{0.4, 0.6, 0.8\}$. EVEN was able to thin its ensembles on most datasets to approximately *one fourth* their original size without any significant degradation in performance.

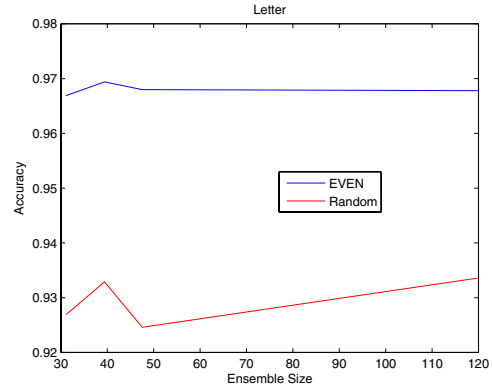


Fig. 2. Letter: The effects of evolutionary thinning, showing accuracy vs ensemble size.

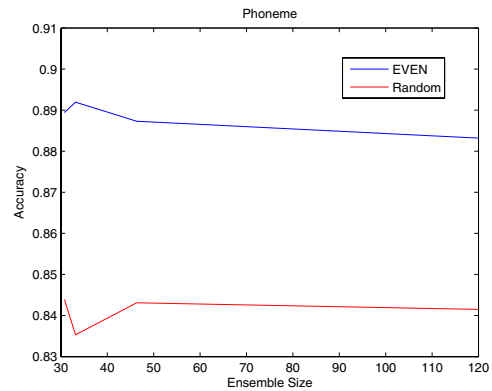


Fig. 3. Phoneme: The effects of evolutionary thinning, showing accuracy vs ensemble size.

The thinned ensembles selected by EVEN outperformed randomly selected ensembles of equal size. For instance, Figures 2 to 3 show the effects of thinning on EVEN accuracy for the Letter and Phoneme datasets. On the x-axis of these graphs is the ensemble size resulting from the cut-off values. Thus, the smallest size is the cut-off value of 0.8 and the largest size is the cut-off value of 0.0 (all classifiers). The graphs thus follow the resulting pairs of size and accuracy from cut-off values of $\{0.8, 0.6, 0.4, 0.0\}$. There is a compelling correlation between the κ values and the corresponding improvement over random selection for thinning. Again, due to space considerations, we only show it for two datasets: one with 26 classes and the other with two classes.

Table V shows the κ values for all the datasets when we applied the cut-off level of 0.8. That is only the classifiers with weights greater than 0.8 were selected. This maintained the performance of the ensemble but significantly reduced the ensemble size. The κ values in the Table show an interesting trend. For the datasets such as letter that noticed a significant improvement on thinning, the κ value is low, suggesting a high diversity in the thinned ensemble. Only for LED we notice a significant increase in the κ value and LED is indeed one of the datasets on which there is little improvement

TABLE II
ACCURACY OF HETEROGENEOUS ENSEMBLES OF 120 CLASSIFIERS.

	DNA	Hypo	KRKP	LED	Letter	Page	Pendigit	Phoneme	Satimage	Sick
EVEN	95.80	97.75	99.25	75.17	96.78	97.24	99.32	88.32	91.09	98.41
EVEN (cut-off = 0.8)	95.66	97.66	99.23	74.97	96.69	97.34	99.34	88.94	91.31	98.36
Majority Vote	95.80	97.77	98.90	74.91	93.36	96.78	99.02	84.15	89.48	98.34
Weighted Vote	95.68	97.79	98.96	74.97	94.21	96.79	99.07	84.96	89.60	98.31
Best classifier	95.18	97.47	99.15	74.77	93.85	96.75	99.11	87.41	89.40	98.36
Stacking NB	95.45	97.28	99.25	75.01	94.50	97.15	99.19	85.07	89.25	98.04
Stacking LR	94.97	97.33	99.25	74.04	96.44	96.93	99.18	88.88	90.28	98.64

provided by EVEN over simple majority voting.

We also plotted κ_D versus accuracy plots [4] for the phoneme dataset, as an example. Figure 4 shows plots of kappa vs accuracy for the ensembles EVEN generated at a cutoff level of .8 for the phoneme dataset. The κ_D plot shows that thinning results in a more oblong shape with a defined spread between κ_D values and accuracy¹.

B. Homogeneous Ensembles – Bagging.

We saw little improvement over majority voting with ensembles of bagged decision trees. We believe that this is due to the low diversity of the bagged trees, as was also observed by Dietterich [4] and Chawla et al. [3]. While we were able to generate a thinned ensemble that performed as well as the entire ensemble, albeit with much fewer classifiers, it did not lead to the performance improvement we observed for the heterogeneous ensembles. Moreover, the random selection of classifiers from the 500 set is able to achieve the same performance generated by EVEN. Nevertheless, EVEN is a more principled approach for thinning.

There is little advantage to using EVEN for ensemble construction because of a lack of large variance among the classifiers. For instance, the diversity results of phoneme dataset in Figures 5 show the tight clustering of points demonstrative of low diversity classifiers.

Because of the low diversity generated by bagged trees, small random subsets of the classifiers were able to perform as well as the full set. To investigate how many classifiers were actually needed out of the 500 which were constructed,

¹For consistency, we will continue show the diversity plots for phoneme dataset with bagging.

TABLE III

NUMBER OF TIMES THERE WAS A STATISTICALLY SIGNIFICANT DIFFERENCE, USING PAIRED T-TEST AT 95% CONFIDENCE, BETWEEN EVEN AND OTHER COMBINATION METHODOLOGIES. IT WAS NEVER SIGNIFICANTLY WORSE.

	Uniform Vote	Weighted Vote	Best Classifier	Stacking (NB)	Stacking (LR)
EVEN	5	5	5	4	4
EVEN (cut-off = 0.8)	6	6	5	4	3

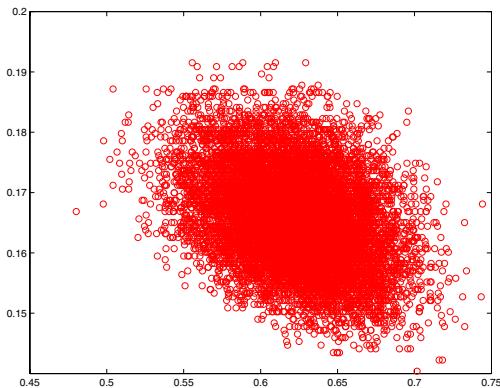


Fig. 5. Phoneme: accuracy vs Kappa for a bagging using the thinned ensemble (cut-off = 0.8).

we tested ensembles ranging in size from 1 to 50 trees. Figures 6 and 7 show the learning curves as more classifiers are added to the ensemble. As expected, the curve levels off very early, showing that less than twenty trees are typically required. The learning curves show that EVEN is able to pick up classifiers that perform well on the validation set, but the performance levels off on the testing set. We would like to conduct a bias and variance analysis of the error to further understand the impact of EVEN on the homogeneous bagged ensembles.

EVEN does not perform well for bagging because of the lack of sufficient variance among the classifiers. The evolving classifier combinations are less sensitive to the addition or deletion of member-classifiers, meaning that a small change in the weight of one classifier is not likely to change the prediction of the entire ensemble because they are so clustered in their predictions. Therefore changes to a generation were more likely to be lost, reducing the ability of the genetic algorithm to make iterative improvements.

V. CONCLUSIONS AND FUTURE WORK

We proposed a genetic algorithm based framework, EVEN, for assigning weights to independently learned classifiers in an ensemble. EVEN essentially encourages collaborative or collective behavior of the members of the ensemble. The fact that accuracy weighted vote does not do as well as the EVEN weighted vote sheds light on the argument that it is the collec-

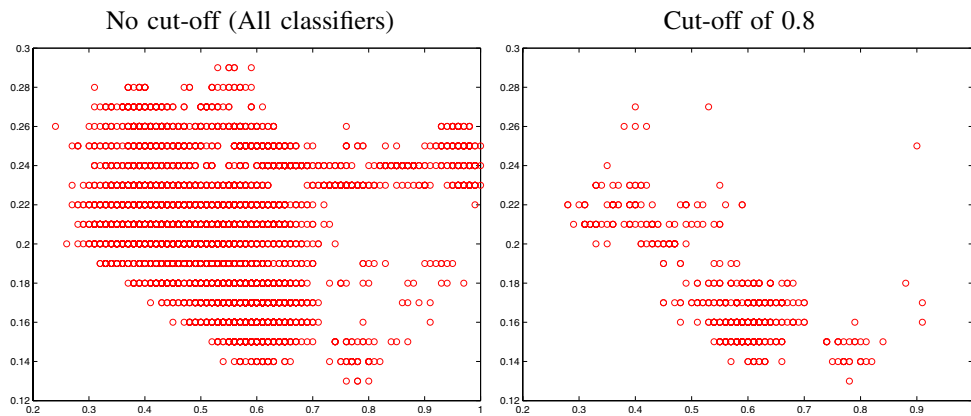


Fig. 4. κ_D plots for the Phoneme dataset using the Heterogeneous Ensemble of size 120.

TABLE IV
COMPARISON OF EVEN AND MAJORITY VOTING FOR HETEROGENEOUS ENSEMBLES OF SIZE 120 AND 132.

	DNA	Hypo	KRKP	LED-25	Letter	Page	Pendigit	Phoneme	Satimage	Sick
EVEN 120	95.80	97.75	99.24	75.01	96.78	97.24	99.32	88.32	91.09	98.41
EVEN 132	94.26	97.71	99.54	75.12	96.58	97.94	99.43	86.49	91.73	97.51
Voting 120	95.80	97.77	98.90	74.91	93.36	96.78	99.02	84.15	89.48	98.34
Voting 132	94.15	97.68	99.38	74.67	93.20	97.20	98.85	83.00	89.34	97.71

TABLE V
 κ VALUES FOR THE 120 HETEROGENEOUS ENSEMBLES AND THE THINNED ENSEMBLE AT CUT-OFF OF 0.8.

cutoff	dna	hypo	krkp	led	letter	page	pen	pho	sat	sick
0.0	.2732	.3634	.0791	.3610	.0831	.4110	.0940	.4314	.2945	.3869
0.8	.2782	.4176	.0787	.4557	.0664	.4091	.0523	.3865	.3008	.3542

TABLE VI
ACCURACY OF BAGGED DECISION TREES. THE FIGURES IN PARENTHESES ARE THE NUMBER OF CLASSIFIERS WHICH WERE SELECTED BY EVEN OUT OF THE FULL 500 TREES.

	DNA	Hypo	KRKP	LED-24	Letter	Page	Pendigit	Phoneme	Satimage	Sick
EVEN, no cut	93.11	97.89	99.17	74.27	93.10	97.57	98.36	88.05	90.99	98.89
EVEN, cut = .8	93.32	97.89	99.17	73.47	93.17	97.69	98.48	87.93	90.99	98.59
	(119)	(93)	(106)	(138)	(149)	(119)	(140)	(148)	(168)	(93)
Majority Vote	93.11	97.89	99.17	74.53	92.83	97.57	98.36	87.81	90.58	98.77
Weighted Vote	93.11	97.89	99.17	74.67	92.87	97.57	98.36	87.81	90.48	98.77
Best Classifier	92.07	97.47	100.0	62.80	83.67	97.20	95.76	84.24	86.13	98.24

tive behavior of the members of an ensemble (canceling out each other's error) rather than individual high performances that lead to improvement. We also implemented thinning of an ensemble using a cut-off value on the weights generated by EVEN. It is remarkable that using weights generated by EVEN, we are able to significantly reduce the classifier size and still result in similar performance as using the entire EVEN weighted ensemble. This is again indicative of the fact that we are more interested in a set of classifiers that complement each other. The entire ensemble of 120 or 132 classifiers can have a redundancy in its predictions, and by selecting a smaller subset we trade that redundancy in favor

of collective predictions.

EVEN was an improvement over voting for eight of ten datasets when combined with heterogeneous ensembles. Four of these wins were statistically significant, by using a paired t-test at 95% confidence level. A key advantage of using EVEN for heterogeneous classifiers is that one can potentially learn a library of models on a variety of datasets. Each classifier, with its own inductive bias, may perform better on one or the other datasets and none all. One can then let EVEN search for the optimal weights for those classifiers and identify the best performing set, leading to overall performance improvement. EVEN also

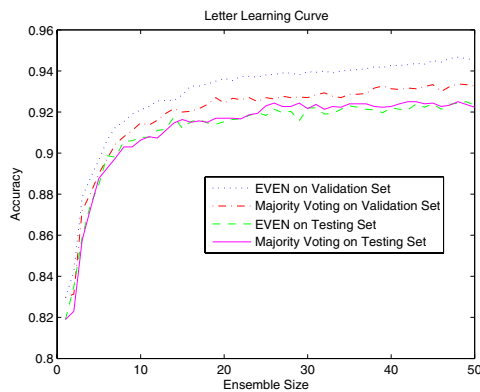


Fig. 6. Letter: Learning curve for ensembles of bagged decision trees.

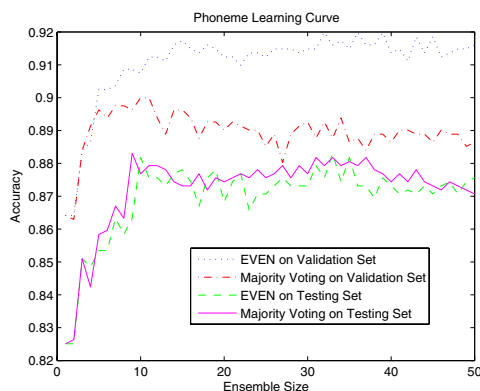


Fig. 7. Phoneme: Learning curve for ensembles of bagged decision trees.

outperformed the stacking approach using Naïve Bayes and Logistic Regression. EVEN's performance is dependent on having a set of diverse classifiers to build upon. This was evident in the experiments conducted with bagged trees. Because there was little diversity between the trees, EVEN was unable to improve upon a simple vote. We are also interested in conducting a thorough bias-variance analysis of the EVEN weighted/selected classifiers.

It would also be interesting to test EVEN when combined with boosting. Boosting produces more diverse classifiers than bagging and leads to a reduction in both the bias and variance components of the error. Since boosting already incorporates a set of weights into the classifiers it produces, comparison could be drawn between the weights created by boosting and those learned by EVEN. Furthermore, the boosting weights could be used to seed the initial population within EVEN, instead of starting with a random population. EVEN could also be easily redesigned as a thinning system to more explicitly optimize towards diversity instead of whatever the performance metric is that is being used.

ACKNOWLEDGEMENTS

We are thankful to David Cieslak for the helpful review of the paper.

REFERENCES

- [1] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
- [2] L. Breiman, "Pasting bites together for prediction in large data sets," *Machine Learning*, vol. 36, no. 1,2, pp. 85–103, 1999.
- [3] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Learning Ensembles From Bites: A Scalable and Accurate Approach," *Journal of Machine Learning Research*, vol. 5, pp. 421 – 451, 2004.
- [4] T. Dietterich, "An empirical comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139 – 157, 2000.
- [5] T. G. Dietterich, "Ensemble methods in machine learning," *Lecture Notes in Computer Science*, vol. 1857, pp. 1–15, 2000.
- [6] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Thirteenth International Conference on Machine Learning*, 1996.
- [7] K. Woods, W. P. Kegelmeyer, and K. W. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. 19, pp. 405–410, 1997.
- [8] 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.
- [9] L. Kuncheva, "Diversity in multiple classifier systems," *Information Fusion*, vol. 6, pp. 2–3, 2005.
- [10] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A new ensemble diversity measure applied to thinning ensembles," in *4th International Workshop on Multiple Classifier Systems*, pp. 306–316, 2003.
- [11] S. Forrest, "Genetic algorithms," *ACM Computing Surveys*, vol. 28, pp. 77–80, 1996.
- [12] D. Opitz, "Feature selection for ensembles," in *AAAI/IAAI*, pp. 379–384, 1999.
- [13] 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.
- [14] 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.
- [15] Y. Kim, N. Street, and F. Menczer, "Meta-evolutionary ensembles," in *IEEE Intl. Joint Conf. on Neural Networks*, pp. 2791–2796, 2002.
- [16] 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.
- [17] 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.
- [18] L. I. Kuncheva and L. C. Jain, "Designing classifier fusion systems by genetic algorithms," *IEEE-EC*, vol. 4, p. 327, November 2000.
- [19] D. Caragea, A. Silvescu, and V. Honavar, "Analysis and synthesis of agents that learn from distributed dynamic data sources," *Lecture Notes in Computer Science*, vol. 2036, pp. 547–561, 2001.
- [20] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.
- [21] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: The MIT Press, 1992.
- [22] C. Reeves and J. Rowe, *Genetic Algorithms - Principles and Perspectives: A Guide to GA Theory*. MA: Kluwer, 2003.
- [23] M. Wall, "Galib: A c++ library of genetic algorithm components. (version 2.4, revision b)," 1998.
- [24] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools with Java implementations*. San Francisco, CA: Morgan Kaufmann, 2000.
- [25] J. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1992.
- [26] C. Blake and C. Merz, "UCI repository of machine learning databases," 1998.
- [27] J. Sylvester and N. V. Chawla, "Evolutionary ensembles: Combining learning agents using genetic algorithms," in *AAAI Workshop on Multiagent Learning*, pp. 46–51, 2005.
- [28] A. Fleiss, *Statistical methods for rates and proportions*. John Wiley and Sons, 1981.