A Novel Ensemble Selection Technique For Weak Classifiers

Kung-Hua Chang[1], **and D. Stott Parker**[1]
[1]University of California Los Angeles
Los Angeles, CA, USA
{kunghua,stott}@cs.ucla.edu

Abstract - *Over the past decade ensemble selection has been proposed as an "overproduce and select" method for constructing ensemble classifiers from simpler individual classifiers. Many prior research papers suggest using the top performing 10%-20% of classifiers in an ensemble. In this paper, we simulate a duel between the top performing (strong) X% of classifiers and the bottom performing (100−X)% (e.g. the top 20% versus the bottom 80%). We propose an ensemble selection algorithm that can effectively use them to construct much stronger classifiers, and apply the algorithm to find the best ensemble (of top performing classifiers as well as of bottom performing classifiers). We also show that using the bottom performing classifiers can yield comparable and sometimes better performance. Furthermore the bottom classifiers can outperform top classifiers for many different values of X, and in some cases all values of X. Our algorithm is based on heuristic search algorithms for developing ensembles of diverse classifiers that optimize complementarity. These results are based on experiments made with 6 publicly available datasets and heterogeneous ensembles using 22 kinds of classifiers*.

Keywords: Ensemble Selection

1 Introduction

Ensemble methods have been shown both theoretically and empirically to outperform individual classification methods in a wide variety of settings and datasets [2][3]. For example, the winners [19][20] of the Netflix Challenge [18] used ensemble methods. Basically, in selecting an ensemble of classifiers, it is common practice to limit the candidates from classifiers with high performance under some evaluation metrics. This approach makes intuitive sense and generally delivers performance improvements. This paper was inspired specifically by experience with the strategy proposed in [3][4][5] of forming ensembles by selecting among only the top 10% of models yielding greatest accuracy. We began to wonder whether greater representation of models with lower accuracy scores might be beneficial. In our experience, strong classifiers can often make the same wrong classifications, particularly on minority examples and classes (because they often sacrifice minority examples/classes and embrace majority examples/classes).

As a result, ensembles built from strong classifiers often show limited improvement in classification accuracy. In order to improve this accuracy, ensembles must maintain a certain level of classifier diversity so as to avoid making the wrong classification altogether. That is, ensembles must have classifiers that can correctly classify minority examples/classes to offset the mistakes made by stronger classifiers. However, real improvement usually requires a high level of diversity – which we formalize as complementarity – that can be hard to produce in the first place. Thus, we need to understand the relationship between diversity, classification accuracy from individual classifiers, and the performance of ensembles. A premise behind this paper is that optimizing complementarity can benefit ensembles. In this paper we develop the idea of maximal complementary ensembles. “Complementary” here refers to the idea that in a teamwork environment, we usually do not put people with the same skills on the same team, but instead put together an ensemble using people with different skills as a way to diversify weaknesses. “Maximal complementary ensembles” mean that we search heuristically and explicitly for a minimal ensemble having maximally diverse classifiers..

2 Related Work

Data mining research usually considers only models that are optimal under some criterion. Choosing the top performing models has a history of success, but it has also introduces many serious problems, including overfitting and bias. A great deal of research in ensemble methods [1-6,13-16] has been aimed at these problems. Throughout this research, diversity has been recognized to be important in improving ensemble performance. Many measures of diversity have been considered in the literature for ensemble methods; cf. [16]. For example [17] suggested that an ideal ensemble consists of highly correct classifiers that disagree as much as possible. The use of "maximum diversity" was considered in [14], as a kind of generalized diversity seeking to develop ensembles in which incorrect labeling by one classifier is countered by correct labeling by another. Krogh & Vedelsby’s prior work [15] showed that ensemble error is directly related to the average accuracy of the ensemble plus a term measuring diversity (called ambiguity in the original paper). This particular property will be used in our paper to justify our algorithm.

3 Algorithms

3.1 Background

The heart of our strategy is to select excellent ensembles of classifiers from a large and diverse pool. These teams are deliberately kept as complementary as possible. By complementary here we mean that classifiers are selected incrementally so as to cover any remaining incorrectly handled cases in the training set. This can be done by selecting minimal sets of team members that correctly classify as many cases in the training set as possible. We employ the simplest classifier combination method – majority voting – and choose individual classifiers based on their training accuracy and on their misclassifications on the training set.

To communicate the basic idea let us limit our discussion to the simplest scenario: in majority voting during ensemble selection, if we have 1 incorrect vote (misclassification), we need 2 correct votes (correct classifications) to compensate. That is, if we have N bad votes, we need (N+1) correct votes to obtain the correct prediction outcome. Thus, a problem occurs when combining multiple classifiers together in which a majority cast incorrect votes. In order to offset them, we need to find enough classifiers to cast correct votes. In other words, the resulting ensemble will have at least N + (N+1) = 2N+1 classifiers. We can see that if we can reduce the number of incorrect votes at a much earlier stage, then we can greatly reduce the number of classifiers N needed. Intuitively, our strategy will be to identify where these incorrect votes are distributed while we are in the early stage of finding classifiers to add to the ensemble. We can then fill in correct votes accordingly in these soft spots to improve performance of the ensemble. However, the question is: how do we discover the way these incorrect votes are distributed?

3.2 Maximal Complementary Ensembles

Krogh & Vedelsby [15] proposed that ensemble error consist of the generalization errors of the individual classifiers plus a term measuring diversity (called ambiguity in the original paper). We will use representation from (Opitz and Shavlik, 1996) [17] below:

$$\hat{E} = \overline{E} - \overline{D}$$

Where $\overline{E} = \sum_i w_i E_i$ is the weighted average of the individual classifier's generalization error, and $\overline{D} = \sum_i w_i D_i$ is the weighted average of the diversity among these classifiers. (Opitz and Shavlik, 1996) [17] suggested that an ideal ensemble consists of highly correct classifiers that disagree as much as possible. If we want to have a near-perfect ensemble, we will need to have the ensemble generalization error as close to zero as possible. That is, in order to achieve

$$\hat{E} = \overline{E} - \overline{D} = 0$$

we must have:

$$\overline{E} = \overline{D}$$

$$\sum_i w_i E_i = \sum_i w_i D_i$$

In majority voting, if no individual classifier can predict everything correctly (that is, it has its own generalization error), then we need at least 3 classifiers (because 2 correct votes are needed for each incorrect vote). So we can divide $\hat{E}$ as $\hat{E}_1$, $\hat{E}_2$, and $\hat{E}_3$ where $\hat{E}_1$ is the weighted average of the generalization error of the first group of ensemble (called ENS_1), and $\hat{E}_2$ is the weighted average of the generalization error of the second group of ensemble (called ENS_2). $\hat{E}_3$ is simply an individual classifier's generalization error and we can name it as ENS_3 because a single classifier can form an ensemble by itself, so

$$\hat{E}_3 = \overline{E_3} - \overline{D_3}$$

Then we seek to achieve

$$\hat{E} = \hat{E}_1 + \hat{E}_2 + \hat{E}_3 = 0$$

$$\hat{E}_1 + \hat{E}_2 = -\hat{E}_3$$

$$\hat{E}_1 + \hat{E}_2 = \overline{D}_3 - \overline{E}_3$$

$$\hat{E}_1 + \hat{E}_2 + \overline{E}_3 = \overline{D}_3$$

This means that when we add the final individual classifier into the ensemble, we hope to have the combined averaged ensemble generalization error as close to the final individual classifier's diversity (the difference between ENS_{12} and ENS_3) as possible. We can discuss possible scenarios below.

Suppose we do a majority voting on the first and second groups of the ensemble to form a new ensemble (called ENS_{12}) having generalization error $\hat{E}_{12}$.

$$\hat{E}_{12} = \hat{E}_1 + \hat{E}_2$$

Then assuming majority voting, an ideal situation will be:

$$\hat{E}_{12} = \overline{D}_3 - \overline{E}_3$$

At this point, $\overline{D}_3$ is the diversity between ENS_{12} and ENS_3. Figure 1 shows the ideal scenario when we select the final classifier in our ensemble.

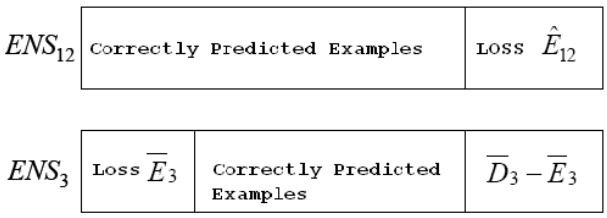


Figure 1. Ideal scenarios when measuring losses (incorrect predictions) in the training set. We intentionally display the losses as grouped together to simplify the presentation.

Since $\overline{D}_3$ is the diversity between ENS_{12} and ENS_3, $\overline{D}_3 - \overline{E}_3$ is a set that differs from the losses ($\hat{E}_{12}$) in ENS_{12}.

That means $\overline{D}_3 - \overline{E}_3$ is a set that can cast correct votes for the final ensemble. Besides, ideally the losses $\overline{E}_3$ should have no impact at all on the final ensemble because there should exist enough correct votes in ENS_{12} such that the incorrect votes from $\overline{E}_3$ will be corrected. Figures 2, 3, and 4 are examples that illustrate the ideal situation:

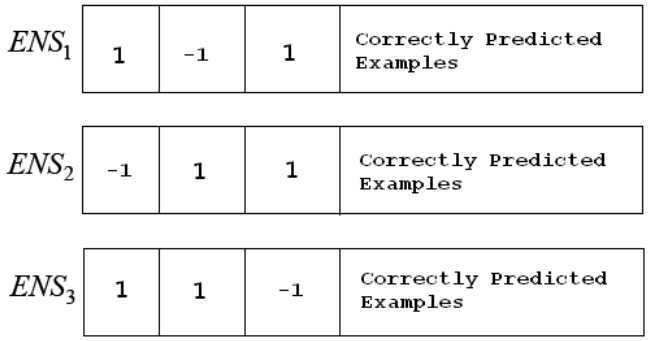


Figure 2. In this scenario, "correctly predicted examples" represent training examples for which there are sufficiently many correct votes, and 1 incorrect vote cannot change the outcome. "1" means currently we have one more correct vote than incorrect vote. "-1" means that we have one more incorrect vote than correct votes.

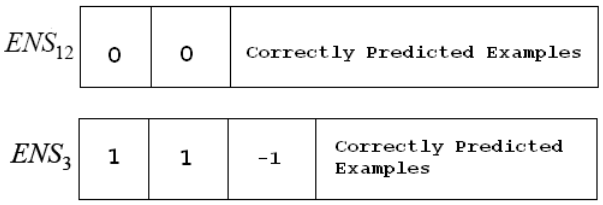


Figure 3. After we perform majority voting on the first and second ensemble, we can see that ENS_{12} has 2 places where 0 exists (0 means indecision due to equal number of 1 and -1); these places have the same number of correct and incorrect votes.

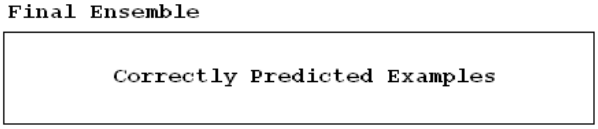


Figure 4. After we perform majority vote on ENS_{12} and ENS_3, we obtain a perfect ensemble.

An interesting phenomenon arises when we are adding the last classifier into our ensemble as illustrated in Figure 5.

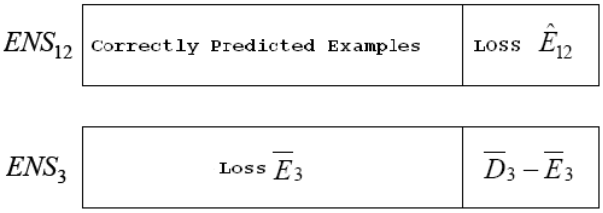


Figure 5. Problematic scenario in constructing a perfect ensemble.

If $\overline{E}_3$ is quite large, then this means ENS_3 is an ensemble (classifier) having very poor performance (we call it weak). However, this weak ensemble (classifier) can help our algorithm construct a perfect ensemble. This finding differs from what (Caruana et al., 2006) proposed, which was to set pruning levels only among the top 10-20% of models (classifiers).

(Krogh & Vedelsby, 1995) showed that "the generalization error of the ensemble is always smaller than the (weighted) average of the ensemble errors: $E < \overline{E}$." In particular, for uniform weights:

$$E \le \frac{1}{N} \sum_\alpha E^\alpha$$

Thus, if $\sum_\alpha E^\alpha = \sum_\alpha D^\alpha$, then $E = 0$.

3.3 Ensemble Selection Algorithm

We can construct an algorithm by extending the relationship between ENS_{12} and ENS_3 such that ENS_{12} can be an ensemble having only one classifier while ENS_3 is the next classifier we want to add into our ensemble. The algorithm will continue to add in new classifier if it satisfies certain conditions (described later) until either we have a perfect ensemble or we cannot improve performance further (beyond some predefined threshold). The algorithm will use the training set to measure diversity by comparing the differences of how well ENS_{12} and ENS_3 do on the training set. (Please note that the algorithm uses selection with replacement. That is, we allow a classifier to be added to the ensemble multiple times.)

The cost function is similar to a 0-1 loss function. If a classifier correctly predicts the true label for one example in training set, then we label it +1. Otherwise, we assign a -1 to it. Thus a classifier can be viewed as a set consisting of +1 and -1, and majority voting adds up the values from different classifiers. We can see that in an ensemble, a value of 0 reflects indecision due to the same number of +1 and -1 values.

Since the performance optimization problem is hard, we will propose an approximation algorithm. Rather than find exact matches of $\hat{E}_{12}$ and $\overline{D}_3 - \overline{E}_3$. We can instead add a classifier to the ensemble with $\overline{D}_3 - \overline{E}_3$ as close to $\hat{E}_{12}$ as possible.

Notice however that if one classifier is the complete opposite of another, as illustrated in Figure 6, then it is useless to perform majority voting with these 2 classifiers.

ENS_{12}	-1	-1	1
ENS_3	1	1	-1

Figure 6. Two classifiers that are complete opposites.

Figure 7 shows that another condition needed is for $\overline{E}_3$ to minimize damage to $(\overline{D}_{12} - \overline{E}_{12})$:

ENS_{12}	$\overline{D}_{12} - \overline{E}_{12}$	Correctly Predicted Examples	Loss $\overline{E}_{12}$
ENS_3	Loss $\overline{E}_3$	Correctly Predicted Examples	$\overline{D}_3 - \overline{E}_3$

Figure 7. Relationship between ENS_{12} and ENS_3.

Algorithm 1 (shown in Figure 8) always chooses to add a classifier that can maximally correct the incorrectly classified instances in the current ensemble, while minimizing the damage the new classifier brings to the ensemble. Since the algorithm always corrects incorrect votes at the earliest possible stage, the total number of classifiers needed in an ensemble can be greatly reduced. Besides, since it always chooses a new classifier that differs most from the ensemble, it eliminates redundancy in the classifiers it selects.

The search method employed here is best-first search. However, it is computationally costly when the search depth is large (e.g. more than 30). So we adopt an alternative that searches both leaf nodes (the last classifiers added to the ensemble) and also next-to-root nodes (individual classifiers that are added to our ensemble second, as shown in Algorithm 1 as the set C). One reason is that it is infeasible to perform complete depth-first search in leaf nodes, and the other reason is that a next-to-root node usually has greater impact on ensemble selection than a leaf node. In other words, when we are selecting the last individual classifier (leaf node) for our ensemble, most of the votes are established and the last vote often has little effect on the outcome of the final ensemble. But the second selection (next-to-root node) can sometimes greatly change the outcome and can lead to very different selections in later individual classifiers.

3.4 Examples

Suppose we have a classification problem given training set T = [1, 2, 3, 1, 2, 3], which is a 3-class classification problem with the following 5 classifiers:

C1 = [1, 2, 2, 1, 2, 2] Training Accuracy = 4/6
C2 = [2, 2, 3, 2, 2, 3] Training Accuracy = 4/6
C3 = [3, 3, 3, 1, 3, 3] Training Accuracy = 3/6
C4 = [1, 1, 2, 1, 2, 1] Training Accuracy = 3/6
C5 = [1, 1, 1, 1, 1, 1] Training Accuracy = 2/6

We first transform the classifiers C1 to C5 (as G1 to G5) using the cost function we proposed as follows:

Algorithm 1 Maximal Complementary Ensemble

1: **Input**: M classifiers
2: **For** i = 1 to M
3: **Do;**
4: Include the i-th classifier in initial ensemble set α .
5. **For** j = 1 to M
6: **Do;**
7: A = $\arg\max_j((D_j - E_j) \cap E_\alpha)$
8: B = $\arg\min_j((D_\alpha - E_\alpha) \cap E_j)$
9: C = $A \cap B$
10: **End;**
11: **For** j = 1 to the number of classifiers in C
12: **Do;**
13: Save α in a temporary set.
14: add j-th classifier from C into α with majority vote.
15: Threshold = 1
16: **Repeat**
18: A2 = $\arg\max_j((D_j - E_j) \cap E_\alpha)$
19: B2 = $\arg\min_j((D_\alpha - E_\alpha) \cap E_j)$
20: C2 = $A \cap B$
21: Add the best classifier from C2 to α with majority vote
22: Record the performance of α (training/test)
23: **If** the performance of α is improved **then**
24: Threshold = 0
25: **Else**
26: Threshold = Threshold + 1
27: **Until** we have a perfect ensemble or we cannot improve its performance after Threshold exceeds 10.
28: Restore α from the temporary set.
29: **End;**
30: **End;**

G1 = [1, 1, -1, 1, 1, -1]
G2 = [-1, 1, 1, -1, 1, 1]
G3 = [-1, -1, 1, 1, -1, 1]
G4 = [1, -1, -1, 1, 1, -1]
G5 = [1, -1, -1, 1, -1, -1]

Suppose we include G1 in the initial ensemble set α, so that $\alpha = \{G1\}$. The set A will choose classifiers 2 and 3 as possible classifiers to add to the ensemble.

$((D_1 - E_1) \cap E_\alpha)$ = [0, 0, 0, 0, 0, 0] Gain: 0
$((D_2 - E_2) \cap E_\alpha)$ = [0, 0, 1, 0, 0, 1] Gain: 2
$((D_3 - E_3) \cap E_\alpha)$ = [0, 0, 1, 0, 0, 1] Gain: 2
$((D_4 - E_4) \cap E_\alpha)$ = [0, 0, 0, 0, 0, 0] Gain: 0
$((D_5 - E_5) \cap E_\alpha)$ = [0, 0, 0, 0, 0, 0] Gain: 0

The set B will choose classifier 2 as possible classifiers to add to the ensemble.

$((D_\alpha - E_\alpha) \cap E_1)$ = [0, 0, -1, 0, 0, -1] Damage: 2
$((D_\alpha - E_\alpha) \cap E_2)$ = [-1, 0, 0, -1, 0, 0] Damage: 2
$((D_\alpha - E_\alpha) \cap E_3)$ = [-1, -1, 0, 0, -1, 0] Damage: 3
$((D_\alpha - E_\alpha) \cap E_4)$ = [0, -1, -1, 0, 0, -1] Damage: 3
$((D_\alpha - E_\alpha) \cap E_5)$ = [0, -1, -1, 0, -1, -1] Damage: 4

Since $C = A \cap B$, classifier 2 will be the only classifier included in set C. Thus, the algorithm will choose C2 to add to the ensemble set α and perform a majority vote, yielding α = [0, 1, 0, 0, 1, 0]. Here '0' means indecision due to equal number of correct and incorrect votes.

The algorithm continues this process until it hits the termination condition. The solution of this example is [C1, C2, C3, C5, C1, C3] such that

C1 = [1, 2, 2, 1, 2, 2]
C2 = [2, 2, 3, 2, 2, 3]
C3 = [3, 3, 3, 1, 3, 3]
C5 = [1, 1, 1, 1, 1, 1]
C1 = [1, 2, 2, 1, 2, 2]
C3 = [3, 3, 3, 1, 3, 3].

Here Majority_Vote of (C1, C2, C3, C5, C1, C3) = [1, 2, 3, 1, 2, 3]. So the training accuracy for the ensemble of these 6 classifiers is 100%, and test accuracy can simply be calculated accordingly. We can see that the weak classifier C5 does help bring the ensemble to 100% training accuracy.

4 Experimental Setup & Results

Table 1 shows the UCI KDD datasets [8] used in the experiment. All attributes in the datasets contain numeric values.

Dataset	#Training	#Test	# Attributes	#Class
balance-scale	417	208	4	3
bupa	230	115	6	2
iono-sphere	234	117	34	2
lung cancer	22	10	56	3
lymp	98	50	18	4
pima	512	256	8	2

Table 1. 6 UCI KDD datasets used in the experiment

4.1 Classifiers

For each dataset, we generated 500 different bagging results and applied 22 different kinds of classifiers (18 from Weka [11] and 4 from LIBSVM [7]) to them to overproduce enough classifiers. The kinds of classifiers considered were: NaiveBayesMultinomial, ComplementNaiveBayes, NaiveBayes, SMO, Logistic, Multilayer Perceptron, AdaBoostM1, LogitBoost, VFI, J48, NBTree, REPTree, RandomForest, ConjunctiveRule, DecisionTable, JRip, PART, Ridor, SVM (Linear), SVM (Polynomial), SVM (RBF), SVM (Sigmoid). Thus we overproduced 11000 classifiers and selected the best ensemble from among these 11000 classifiers for each dataset.

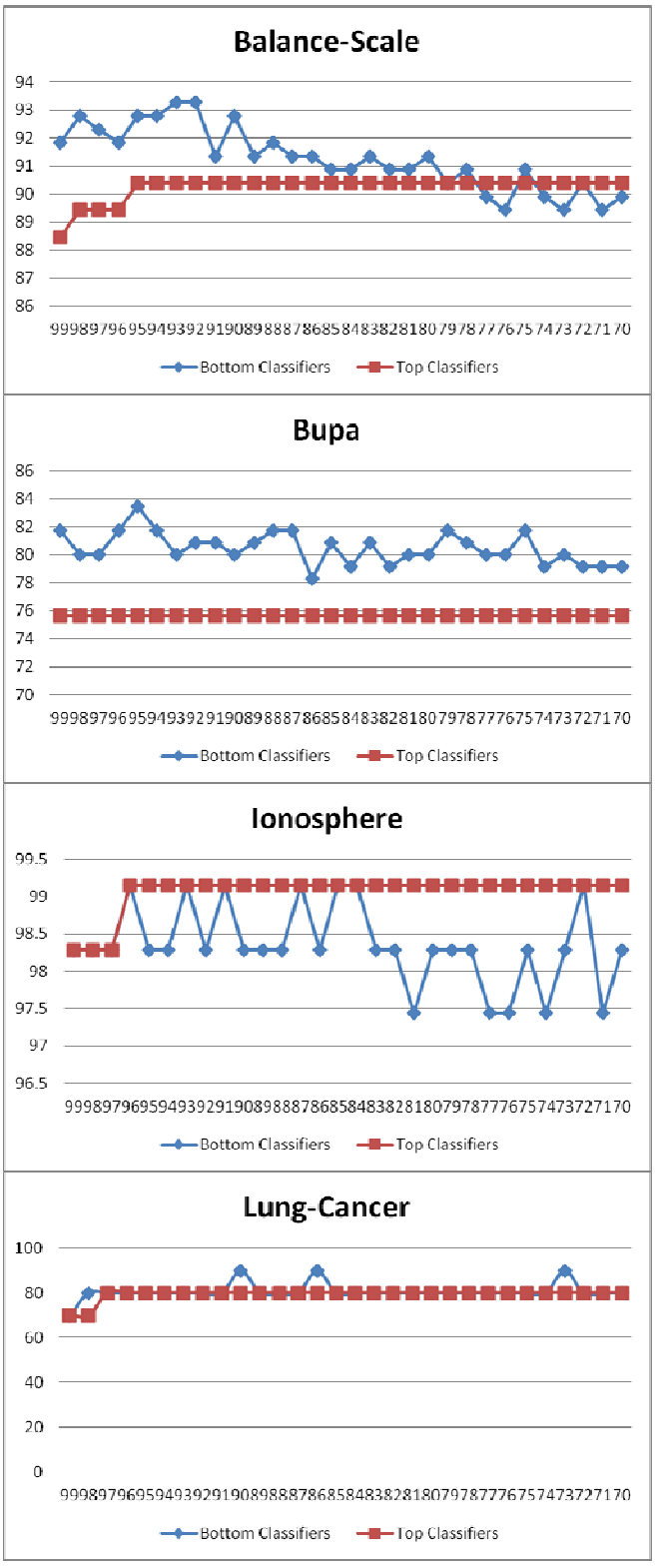
Since an ensemble-based method (e.g. AdaBoostM1 and RandomForest) usually yield better performance than individual learners using a single algorithm, we simply treated these methods as *strong* classifiers. This ensured having both strong and weak classifiers for our experiment.

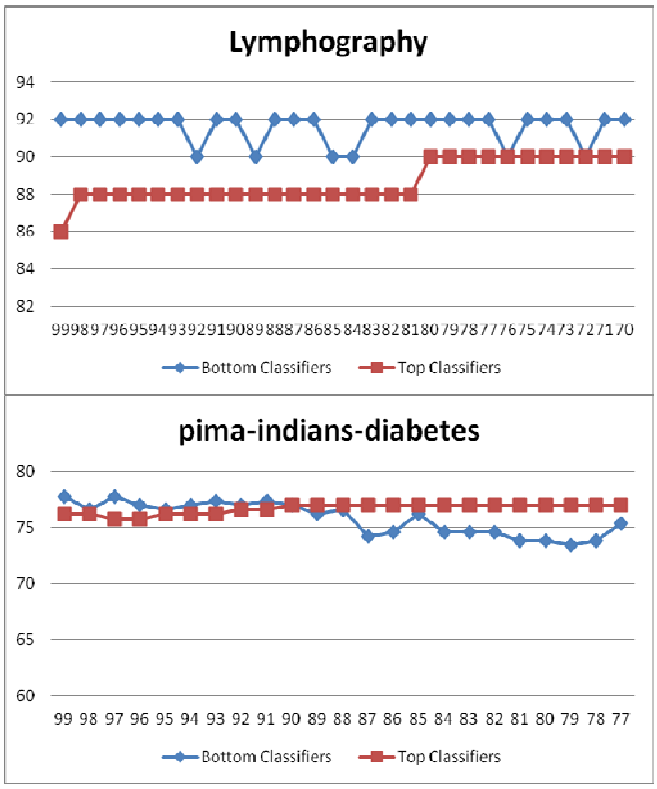
4.2 Experimental Results

We set up the experiments by applying the same ensemble selection algorithm to the top X% of classifiers and compared the result to the remaining (100-X)%. For example, we started the experiment by comparing ensembles from the top 1% and the bottom 99%, then the top 2% versus the bottom 98%, until we reached the top 34% versus the bottom 66%. Below are graphs of test accuracy for all 6 datasets. The experimental results show consistently that bottom classifiers can yield comparable and sometimes better performance than top classifiers.

In the Balance-scale dataset, the bottom classifiers outperformed top classifiers for X values up to 79%. That is, we can discard the top 21% of classifiers and can still make an ensemble from the bottom 79% with better performance. In the Bupa dataset, the outperformance of bottom classifiers continued for the the entire range of X values, from 1% to 34%.

In the Ionosphere, and Lung cancer datasets, ensembles using only bottom classifiers consistently had comparable performance to ensembles using top classifiers. In the Lymphography dataset, bottom classifiers outperformed top classifiers up to 76%.





That is, we could omit the top 24% and still build ensembles with better performance. In the Pima Indian/diabetes dataset, we could reach 90% i.e., ensembles from the bottom 90% outperformed ensembles using the top 10% of classifiers.

It helps to study this phenomenon in detail to appreciate the way in which top performing classifiers can lower performance. For example, in the Balance-Scale dataset, if we use the top 1%-20% of all models, we just cannot achieve the highest possible test accuracy. It seems that the pruned bottom (weaker) classifiers can complement strong classifiers in a way that improves performance. This might seem to be an unusual situation, but in fact the same situation arises in the Bupa, Lymphography, and Pima Indian/Diabetes datasets. This suggests that as long as weaker/strong classifiers are included in the right places, then they are helpful. However, the top X% of classifiers are sometimes sufficient as long as they are complementary. Our experimental results suggest however that bottom (weaker) classifiers can help improve performance in terms of classification accuracy. The full experimental results can be found in our website in [12].

5 Conclusions

In this paper we have explored an ensemble selection strategy that finds complementary ensembles in the construction of ensembles of classifiers, comparing the test accuracy of the top performing X% of classifiers versus the bottom performing (100-X%), and emphasizing diversity in the kinds of classifier considered. The surprising

successfulness of this approach has been explored in the experimental results.

A key aspect of our approach, and a primary contribution of this work, has been in the idea of maximizing diversity while minimizing ensemble size. BBBFS, our heuristic search algorithm, works precisely to limit redundancy among classifiers in an ensemble in this way. The result is a small ensemble of diverse classifiers whose complementarity has been optimized. Caruana et al noted in [3] that "While further work is needed to develop good heuristics for automatically choosing an appropriate pruning level for a data set, simply using the top 10-20% models seems to be a good rule of thumb. An open problem is finding a better pruning method." For example, taking model diversity [1] into account might find better pruned sets. Our algorithm, BBBFS, is a heuristic algorithm for maximizing diversity over the training set, and this approach has given interesting results for Caruana's open problem in the experiments.

Another key aspect of our approach is to highlight the cost of blindly cutting the bottom (weaker) classifiers. Our experiments contradict the validity of using only top classifiers,. We believe that further research is needed to consider factors such as complementarity structure among classifiers and the number of classifiers used in ensembles.

Future work should investigate further why an ensemble of bottom classifiers can sometimes outperform ensembles of top classifiers. We conjecture that overfitting could play a role here, as an ensemble of top classifiers could easily be misled in this way.

6 References

[1] L.I. Kuncheva and C.J. Whitaker, "Measures of diversity in classifier ensembles", Machine Learning 51: 181-207, 2003.

[2] T.G. Dietterich, "Ensemble Methods in Machine Learning", Proc. 1st Intl Workshop on Multiple Classifier Systems, Springer Verlag, LNCS #1857, 1-15, 2000.

[3] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, "Ensemble Selection from Libraries of Models", Proc. Intnl. Conf on Machine Learning, 2004.

[4] R. Caruana, A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms Using Different Performance Metrics", ICML 2005.

[5] R. Caruana, A. Mnson, A. Niculescu-Mizil, "Getting the Most Out of Ensemble Selection", Technical Report 2006-2045, Dept. of Computer Science, Cornell University, 2006.

[6] L. Breiman, "Bagging Predictors", Machine Learning 24(2): 123-140, 1996.

[7] C-C. Chang, C-J. Lin, "LIBSVM : A Library for Support Vector Machines", 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[8] Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[9] lymphography dataset: M. Zwitter, M. Soklic, University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.
http://www.ics.uci.edu/~mlearn/databases/lymphography/lymphography.names

[10] primary-tumor dataset: M. Zwitter, M. Soklic, University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.
http://www.ics.uci.edu/~mlearn/databases/primary-tumor/primarytumor.names

[11] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[12] http://www.cs.ucla.edu/~kunghua/DMIN2013/

[13] L.K. Hansen, P. Salamon, Neural network ensembles. IEEE Trans. Patt. Anal. Mach. Intell. 12(10): 993-1001, 1990.

[14] D. Partridge, W.J. Krzanowski, Software diversity: Practical statistics for its measurement and exploitation. Information & Software Technology, 39, 707–717, 1997.

[15] A. Krogh, J. Vedelsby, Neural Network Ensembles, Cross Validation, and Active Learning, Advances in Neural Information Processing Systems, 231- 238, 1995.

[16] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, 2004.

[17] Opitz, D., and Shavlik, J. Actively searching for an effective neural network ensemble. Connection Science, 8(3/4):337–353, 1996.

[18] J. Bennet and S. Lanning (2007), "The Netflix Prize", www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf.

[19] Y. Koren, "The BellKor Solution to the Netflix Grand Prize", (2009).

[20] A. Töscher, M. Jahrer, R. Bell, "The BigChaos Solution to the Netflix Grand Prize", (2009).