Getting the Most Out of Ensemble Selection*

Rich Caruana, Art Munson, Alexandru Niculescu-Mizil
Department of Computer Science
Cornell University
Technical Report 2006-2045
{caruana, mmunson, alexn} @cs.cornell.edu

Abstract

We investigate four previously unexplored aspects of ensemble selection, a procedure for building ensembles of classifiers. First we test whether adjusting model predictions to put them on a canonical scale makes the ensembles more effective. Second, we explore the performance of ensemble selection when different amounts of data are available for ensemble hillclimbing. Third, we quantify the benefit of ensemble selection's ability to optimize to arbitrary metrics. Fourth, we study the performance impact of pruning the number of models available for ensemble selection. Based on our results we present improved ensemble selection methods that double the benefit of the original method.

1 Introduction

An ensemble is a collection of classifiers whose predictions are combined with the goal of achieving better performance than the constituent classifiers. A large body of research now exists showing that ensemble learning often increases performance (e.g. bagging [3], boosting [21], stacking [25]).

Recently, **ensemble selection** [7] was proposed as a technique for building ensembles from large collections of diverse classifiers. Ensemble selection employs greedy forward selection to select models to add to the ensemble, a method categorized in the literature as **overproduce and choose** [20]. Compared to previous work, ensemble selection uses *many* more classifiers, allows optimizing to arbitrary performance metrics, and includes refinements to prevent overfitting to the ensemble's training data—a larger problem when selecting from more classifiers.

In this paper we analyze four previously unexplored aspects of ensemble selection. First, we evaluate ensemble selection's performance when all the models are calibrated to place their predictions on a canonical scale. Making calibrated models available to ensemble selection provides significant improvement on probability measures such as squared error and cross-entropy. It appears, however, that calibration does not make ensemble selection itself more effective; most of the benefit results from improvements in the base-level models and not from better ensemble building.

Second, we explore how ensemble selection behaves with varying amounts of training data available for the critical forward selection step. Despite previous refinements to avoid overfitting the data used for ensemble hillclimbing [7], our experiments show that ensemble selection is still prone to overfitting when the hillclimb set is small. This is especially true if their model bagging procedure is not used. Surprisingly, although ensemble selection overfits with small data, reliably picking a single good model is even harder—making ensemble selection more valuable. With enough hillclimbing data (around 5k points), overfitting becomes negligible. Motivated by these results, we present a method for embedding cross-validation *inside* ensemble selection to maximize the amount of hillclimbing data.[1] Cross-validation boosts the performance of ensemble selection, doubling its previously reported benefit. While adding cross-validation to ensemble selection is computationally expensive, it is valuable for domains that require the best possible performance, and for domains in which labeled data is scarce.

Ensemble selection's ability to optimize to any performance metric is an attractive capability of the method that is particularly useful in domains which use non-traditional performance measures such as natural language processing [14]. Because of this, the third aspect we investigate is what benefit, if any, comes from being able to optimize to any metric. Our experiments reinforce the intuition that it is best to optimize to the target performance metric;

*This technical report is an expanded version of a paper accepted at the 2006 International Conference on Data Mining.

[1] This is different from wrapping cross-validation *around* ensemble selection, which would not increase the data available for hillclimbing.

however, they also show that minimizing squared error or cross-entropy frequently yields ensembles with competitive performance—seemingly regardless of the metric.

Fourth, we test ensemble selection's performance when only the best $X\%$ models are available for selection. These experiments confirm our intuition that the potential for overfitting increases with more models. Using only the top 10-20% of the models yields performance better than or equivalent to ensemble selection without this model pruning.

2 Background

In this section we briefly review the ensemble selection procedure first proposed by Caruana et al. [7]. Ensemble selection is an overproduce and select ensemble method carried to an extreme where thousands of models are trained using many different learning methods, and overfitting is moderated by applying several techniques.

In ensemble selection, models are trained using as many learning methods and control parameters as can be applied to the problem. Little or no attempt is made to optimize the performance of the individual models; all models, no matter what their performance, are added to the model library for the problem. The expectation is that some of the models will yield good performance on the problem, either in isolation or in combination with other models, for any reasonable performance metric.

Once the model library is collected, an ensemble is built by selecting from the library the subset of models that yield the best performance on the target optimization metric. Models are selected for inclusion in the ensemble using greedy forward stepwise model selection. The performance of adding a potential model to the ensemble is estimated using a hillclimbing set containing data not used to train the models. At each step ensemble selection adds to the ensemble the model in the library that maximizes the performance of the ensemble to this held-aside hillclimbing data.

When there are thousands of models to select from, the chances of overfitting increase dramatically. Caruana et al. describe two methods to combat overfitting. The first controls how ensembles are initialized. The second performs model bagging—analogous to feature bagging [1, 5]—to reduce the variance of the selection process.

Ensemble Initialization: Instead of starting with an empty ensemble, Caruana et al. suggest initializing ensembles with the N models that have the best uni-model performance on the hillclimb set and performance metric.

Bagged Ensemble Selection: It is well known that feature subset selection (e.g. forward stepwise feature selection) is unstable when there are many relevant features [2]. Ensemble selection is like feature selection, where models are features and model subsets are found by forward stepwise selection. Because of this, ensemble selection also has high variance. Ensemble selection uses bagging *over models* to reduce this variance. Multiple ensembles are built from random subsets of the models, and then averaged together. This is analogous to the feature bagging methods proposed by Bay [1] and Bryll et al. [5] and used in random forests [4].

Another technique used in the original paper is allowing models to be added to the ensemble more than once. This provides two benefits. First, models added multiple times get more weight in the ensemble average. Second, when models are added *without* replacement, ensemble performance deteriorates quickly after the best models have been exhausted because poorer models must then be added. This makes deciding when to stop adding models to the ensemble critical because overshooting the optimal stopping point can yield much worse performance. Selection *with replacement* allows selection to continue adding copies of good models instead of being forced to add inferior models. This, in turn, makes deciding when to stop adding models far less critical. *All of the experiments in this paper use these three techniques.*

3 Methodology

We use all of the learning methods and data sets used by Caruana et al. [7], and all of the performance metrics except CAL (a probability calibration metric) and SAR (a metric that combines accuracy, squared error, and ROC area). In addition, we also train models with logistic regression (LOGREG), naïve bayes (NB), and random forests (RF) [4], and experiment with four additional data sets: MG, CALHOUS, COD, and BACT. All of the data sets are binary classification problems. The learning methods and data sets are described in Appendix A and B, respectively. The performance metrics we study are described in the following subsection.

3.1 Performance Metrics

The eight performance metrics we use can be divided into three groups: threshold metrics, ordering/rank metrics and probability metrics.

The threshold metrics are accuracy (ACC), F-score (FSC) and lift (LFT). For thresholded metrics, it is not important how close a prediction is to a threshold, only if it is above or below threshold. See Giudici [10] for a description of Lift Curves. Usually ACC and FSC have a fixed threshold (we use 0.5). For lift, often a fixed percent, p, of cases are predicted as positive and the rest as negative (we use $p = 25\%$).

The ordering/rank metrics depend only on the ordering of the cases, not the actual predicted values. As long as ordering is preserved, it makes no difference if predicted

Table 1. Performance with and without model calibration. The best score in each column is bolded.

	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN
ES-BOTH	0.920	0.888	0.967	**0.982**	**0.972**	0.964	0.932	**0.944**	**0.946**
ES-PREV	**0.922**	0.893	0.967	0.981	0.966	0.965	0.919	0.932	0.943
ES-NOCAL	0.919	**0.897**	0.967	0.982	0.970	0.965	0.912	0.925	0.942
ES-CAL	0.912	0.847	**0.969**	0.981	0.969	**0.966**	**0.935**	0.940	0.940
BAYESAVG-BOTH	0.893	0.814	0.964	0.978	0.963	0.956	0.918	0.934	0.928
BAYESAVG-CAL	0.889	0.820	0.962	0.977	0.960	0.955	0.912	0.925	0.925
MODSEL-BOTH	0.871	0.861	0.939	0.973	0.948	0.938	0.901	0.916	0.918
BAYESAVG-PREV	0.881	0.789	0.956	0.970	0.956	0.947	0.893	0.911	0.913
MODSEL-PREV	0.872	0.860	0.939	0.973	0.948	0.938	0.879	0.892	0.913
MODSEL-CAL	0.870	0.819	0.943	0.973	0.948	0.940	0.892	0.910	0.912
MODSEL-NOCAL	0.871	0.858	0.939	0.973	0.948	0.938	0.861	0.871	0.907
BAYESAVG-NOCAL	0.875	0.784	0.955	0.968	0.953	0.941	0.874	0.892	0.905

values fall between 0 and 1 or 0.89 and 0.90. These metrics measure how well the positive cases are ordered before negative cases and can be viewed as a summary of model performance across all possible thresholds. The rank metrics we use are area under the ROC curve (ROC), average precision (APR), and precision/recall break even point (BEP). See Provost and Fawcett [19] for a discussion of ROC from a machine learning perspective.

The probability metrics are minimized (in expectation) when the predicted value for each case coincides with the true conditional probability of that case being positive class. The probability metrics are squared error (RMS) and cross-entropy (MXE).

3.2 Comparing Across Performance Metrics

To permit averaging across metrics and problems, performances must be placed on comparable scales. Following Caruana et al. [7] we scale performance for each problem and metric from 0 to 1, where 0 is baseline performance and 1 is the best performance achieved by any model or ensemble. We use the following baseline model: predict p for every case, where p is the percent of positives in the data.

One disadvantage of normalized scores is that recovering a raw performance requires knowing what performances define the top and bottom of the scale, and as new best models are found the top of the scale may change. Note that the normalized scores presented here differ from those reported in Caruana et al. [7] because we are finding better models that shift the top of the scales. The numbers defining the normalized scales can be found in Appendix C.

4 Ensembles of Calibrated Models

Models trained by different learning algorithms do not necessarily "speak the same language". A prediction of 0.14 from a neural net does not necessarily mean the same thing as a prediction of 0.14 from a boosted tree or SVM. Predictions from neural nets often are well-calibrated posterior probabilities, but predictions from SVMs are just normalized distances to the decision surface. Averaging predictions from models that are not on commensurate scales may hurt ensemble performance.

In this section we evaluate the performance of ensemble selection after "translating" all model predictions to the common "language" of well-calibrated posterior probabilities. Learning algorithms such as boosted trees and stumps, SVMs, or naïve bayes have poorly calibrated predictions [15]. A number of methods have been proposed for mapping predictions to posterior probabilities. In this paper we adopt the method Platt developed for SVMs [18], but which also works well for other learning algorithms [15]. Platt's method transforms predictions by passing them through a sigmoid whose parameters are learned on an independent calibration set. In this paper, the ensemble selection hillclimb set is used for calibration as well.

Table 1 shows the performance of ensemble selection (ES), model selection (MODSEL),[2] and Bayesian model averaging (BAYESAVG) [8], with and without calibrated models. Results are shown for four different model libraries: 1) only uncalibrated models (NOCAL), 2) only calibrated models (CAL), 3) both calibrated and uncalibrated models (BOTH), and 4) only SVMs are calibrated, to mimic prior experiments [7] (PREV). Each entry is the average of five folds on each of the eleven problems. The last column shows the mean performance over all eight metrics. Rows are sorted by mean performance.

Comparing results for ensemble selection with and without calibration (ES-CAL and ES-NOCAL), we see that calibrating models improves RMS and MXE (significant at .05) but hurts FSC. There is little difference for LFT, ROC, APR

[2] Model selection chooses the best single model using the hillclimb set.

and BEP. For model selection we see the same trends: calibrated models yield better RMS and MXE and worse FSC. The magnitudes of the differences suggest that most if not all of the improvement in RMS and MXE for ensemble selection with calibrated models is due to having better models in the library rather than from ensemble selection taking advantage of the common scale of the calibrated models. We are not sure why calibration makes FSC performance worse for both MODSEL and ES, but again suspect that the differences between ES-CAL and ES-NOCAL are due to differences in the performance of the base-level models.

Having both calibrated and uncalibrated models in the library (ES-BOTH and MODSEL-BOTH) gives the best of both worlds: it alleviates the problem with FSC while retaining the RMS and MXE improvements. For the rest of the experiments in this paper we use libraries containing both calibrated and uncalibrated models.

Unlike with ensemble selection, using calibrated models for Bayesian model averaging improves performance on all metrics, not just RMS and MXE (significant at .05). With calibrated models, Bayesian averaging outperforms model selection but is still not as good as ensemble selection.

5 Analysis of Training Size

The original ensemble selection paper demonstrated the method's effectiveness using relatively small hillclimbing sets containing 1000 data points. Since the data used for hillclimbing is data taken away from training the individual models, keeping the hillclimb set small is important. Smaller hillclimb sets, however, are easier to overfit to, particularly when there are many models from which to select.

To explore ensemble selection's sensitivity to the size of the hillclimb set, we ran ensemble selection with hillclimb sets containing 100, 250, 500, 1000, 2500, 5000, and 10000 data points. In each run we randomly selected the points for the hillclimb set and used the remainder for the test set. The hyperspectral and medis data sets contained too few points to leave sufficient test sets when using a 10K hillclimbing set and were omitted. Due to time constraints and the cost of generating the learning curves, we only used one random sample at each size and did not repeat the experiment.

Figure 1 shows learning curves for our eight performance measures and their mean. Each graph is an average over 9 problems. The x-axis uses a logscale to better show what happens with small hillclimbing sets. Normalized performance scores are plotted on the y-axis. For comparison, the graphs include the performance achieved by picking the single best model (MODSEL).

Unsurprisingly, the performance achieved with both ensemble selection and model selection using only 100 points for hillclimbing is quite bad. As data increases, both methods do better as they overfit less. Interestingly, ensemble selection is hurt less by a small hillclimbing set than model selection, suggesting that it is less prone to overfitting than model selection. Because of this, the benefit of ensemble selection over the best models appears to be strongest when training data is scarce, a regime [7] did not examine. (They used 5k training data with 1k points held aside for ensemble stepwise selection.) As the size of the hillclimbing sets goes from 1k to 10k, ensemble selection maintains its edge over model selection.

With small hillclimb sets, using bagging with ensemble selection is crucial to getting good performance; without it, mean performance using a 100 point hillclimb set drops from 0.888 to 0.817. In contrast, bagging provides very little if any benefit when a very large hillclimb set is used (more than 5000 points with our data sets).

6 Cross-Validated Ensemble Selection

It is clear from the results in Section 5 that the larger the hillclimb set, the better ensemble selection's performance will be. To maximize the amount of available data, we apply cross-validation to ensemble selection. Simply wrapping cross-validation around ensemble selection, however, will not help because the algorithm will still have just a fraction of the training data available for hillclimbing. Instead, we embed cross-validation within ensemble selection so that all of the training data can be used for the critical ensemble hillclimbing step. Conceptually, the procedure makes *cross-validated models*, then runs ensemble selection the usual way on a library of cross-validated base-level models.

A cross-validated model is created by training a model for each fold *with the same model parameters*. If there are 5 folds, there will be 5 individual models (each trained on 4000 points) that are 'siblings'; these siblings should only differ based on variance due to their different training samples. To make a prediction for a *test* point, a cross-validated model simply averages the predictions made by each of the sibling models. The prediction for a *training* point (that subsequently will be used for ensemble hillclimbing), however, only comes from the individual model that did not see the point during training. In essence, the cross-validated model delegates the prediction responsibility for a point that will be used for hillclimbing to the one sibling model that is not biased for that point.

Selecting a cross-validated model, whether during model selection or ensemble selection, means choosing *all* of the sibling models as a unit. If 5-fold cross-validation is used, selection chooses groups containing 5 sibling models at a time. In this case, when selection adds a cross-validated model to a growing ensemble, it really adds 5 different models of the same model type to the ensemble, each of which receives the same weight in the ensemble average.

We ran ensemble selection with 5-fold cross-validation;

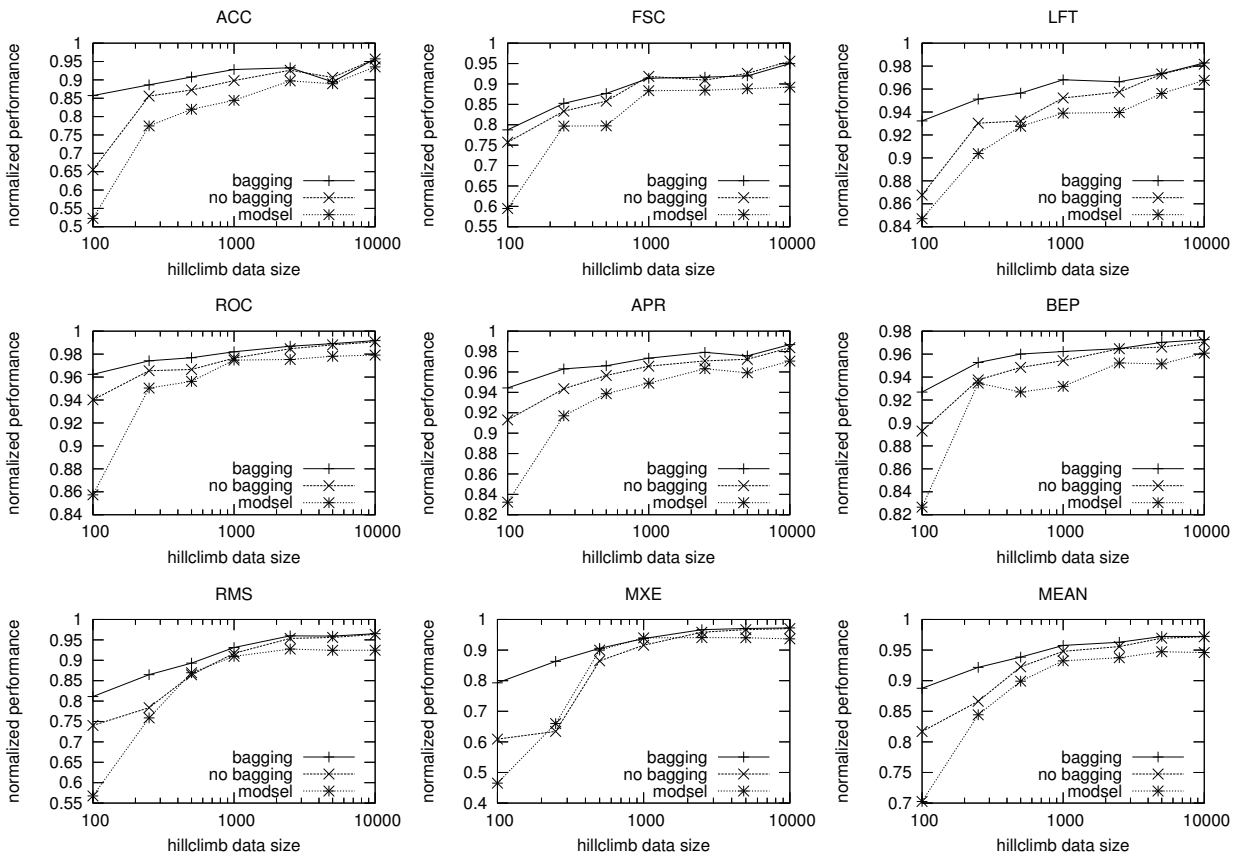

Figure 1. Learning curves for ensemble selection with and without bagging, and for picking the best single model (modsel).

Table 2. Performance with and without cross-validation for ensemble selection and model selection.

	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN
ES-BOTH-CV	**0.935**	**0.926**	**0.982**	**0.996**	**0.992**	**0.977**	**0.984**	**0.989**	**0.973**
MODSEL-BOTH-CV	0.907	0.923	0.971	0.985	0.968	0.963	0.945	0.961	0.953
ES-BOTH	0.920	0.888	0.967	0.982	0.972	0.964	0.932	0.944	0.946
MODSEL-BOTH	0.871	0.861	0.939	0.973	0.948	0.938	0.901	0.916	0.918

this is analogous to normal ensemble selection with a 5000 point hillclimb set. Table 2 shows the results averaged over all the problems. Not only does cross-validation greatly improve ensemble selection performance, it also provides the same benefit to model selection. Five-fold cross-validated model selection actually outperforms non-cross-validated ensemble selection by a small but noticeable amount. However, ensemble selection with embedded cross-validation continues to outperform model selection.

Table 3 provides a different way to look at the results. The numbers in the table (except for the last row) are the percent reduction in loss of cross-validated ensemble selection, relative to non-cross-validated model selection, the baseline used in Caruana et al. [7]. For example, if model selection achieves a raw accuracy score of 90%, and cross-validated ensemble selection achieves 95% accuracy, then the percent reduction in loss is 50%—the loss has been reduced by half. The MEAN row is the average improvement for each metric, across datasets. For comparison, the PREV row is the performance of the original non-cross-validated ensemble selection method (i.e. no cross-validation and only SVMs are calibrated).

Embedding cross-validation within ensemble selection doubles its benefit over simple model selection (from 6.90% to 12.77%). This is somewhat of an unfair comparison; if a cross-validated model library is available, it is just as easy to do cross-validated model selection as it is to do cross-validated ensemble selection. The last row in Table 3 shows

Table 3. Percent loss reduction by dataset.

	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN
ADULT	2.77	5.89	8.72	7.45	6.70	7.58	2.26	4.08	5.68
BACT	2.08	3.83	16.42	4.13	5.49	1.76	1.42	4.15	4.91
CALHOUS	7.95	9.49	48.00	8.69	8.81	6.15	7.17	12.74	13.63
COD	5.73	7.46	14.33	9.14	10.52	7.11	2.39	3.79	7.56
COVTYPE	6.68	7.26	12.35	11.34	14.99	7.64	7.80	12.92	10.12
HS	13.66	16.36	12.32	37.53	37.78	16.77	12.65	27.43	21.81
LETTER.p2	15.21	14.50	100.00	32.84	33.05	15.85	17.13	29.47	32.26
LETTER.p1	21.55	25.66	0.29	69.10	45.29	19.25	19.59	34.58	29.41
MEDIS	2.77	-0.05	2.08	6.33	7.28	4.62	1.40	2.70	3.39
MG	4.45	1.98	4.25	11.84	12.65	6.04	2.57	6.10	6.23
SLAC	2.49	3.27	13.65	6.92	9.62	2.73	1.66	3.33	5.46
MEAN	7.76	8.70	21.13	18.67	17.47	8.68	6.91	12.84	12.77
PREV	4.96	4.56	16.22	8.43	6.24	5.15	3.27	6.39	6.90
MEAN^{cv}	2.89	3.07	10.82	9.97	9.37	2.84	2.54	4.22	5.71

the percent loss reduction of cross-validated ensemble selection compared to cross-validated model selection. Comparing PREV and MEAN^{cv}, we see that after embedding cross-validation, ensemble selection provides *slightly less benefit* over model selection than un-cross-validated ensemble selection did over un-cross validated model selection.

While training five times as many models is computationally expensive, it may be useful for domains where the best possible performance is needed. Potentially more interesting, in domains where labeled data is scarce, cross-validated ensemble selection is attractive because a) it does not require sacrificing part of the training data for hillclimbing, b) it maximizes the size of the hillclimbing set (which Figure 1 shows is critical when hillclimb data is small), and c) training the cross-validated models is much more feasible with smaller training data.

7 Direct Metric Optimization

One interesting feature of ensemble selection is its ability to build an ensemble optimized to an arbitrary metric. To test how much benefit this capability actually provides, we compare ensemble selection that optimizes the target metric with ensemble selection that optimizes a predetermined metric *regardless of the target metric*. For each of the 8 metrics, we train an ensemble that optimizes it and evaluate the performance on all metrics. Optimizing RMS or MXE yields the best results.

Table 4 lists the performance of ensemble selection for a) always optimizing to RMS, b) always optimizing to MXE, and c) optimizing the true target metric (OPTMETRIC). When cross-validation is not used, there is modest benefit to optimizing to the target metric. With cross-validation, however, the benefit from optimizing to the target metric is significantly smaller.

Table 4. Performance of ensemble selection when forced to optimize to one set metric.

	RMS	MXE	OPTMETRIC
ES-BOTH-CV	0.969	0.968	0.973
ES-BOTH	0.935	0.936	0.946

The scatter plots in Figure 2 plot the performance of optimizing to RMS against the performance of optimizing to OPTMETRIC, with one graph per target metric. Again, we can see that ensemble selection performs somewhat better with OPTMETRIC. Always optimizing RMS is frequently very competitive, especially when performance gets close to a normalized score of 1. This is why the benefit of direct metric optimization is so small for cross-validated ensemble selection. These results suggest that optimizing RMS (or MXE) may be a good alternative if the target metric is too expensive to use for hillclimbing.

8 Model Library Pruning

Including a large number of base level models, with a wide variety of parameter settings, in the model library helps ensure that at least some of the models will have good performance regardless of the metric optimized. At the same time, increasing the number of available models also increases the risk of overfitting the hillclimb set. Moreover, some of the models have such poor performance that they are unlikely to be useful for any metric one would want to optimize. Eliminating these models should not hurt performance, and might help.

In this section we investigate ensemble selection's per-

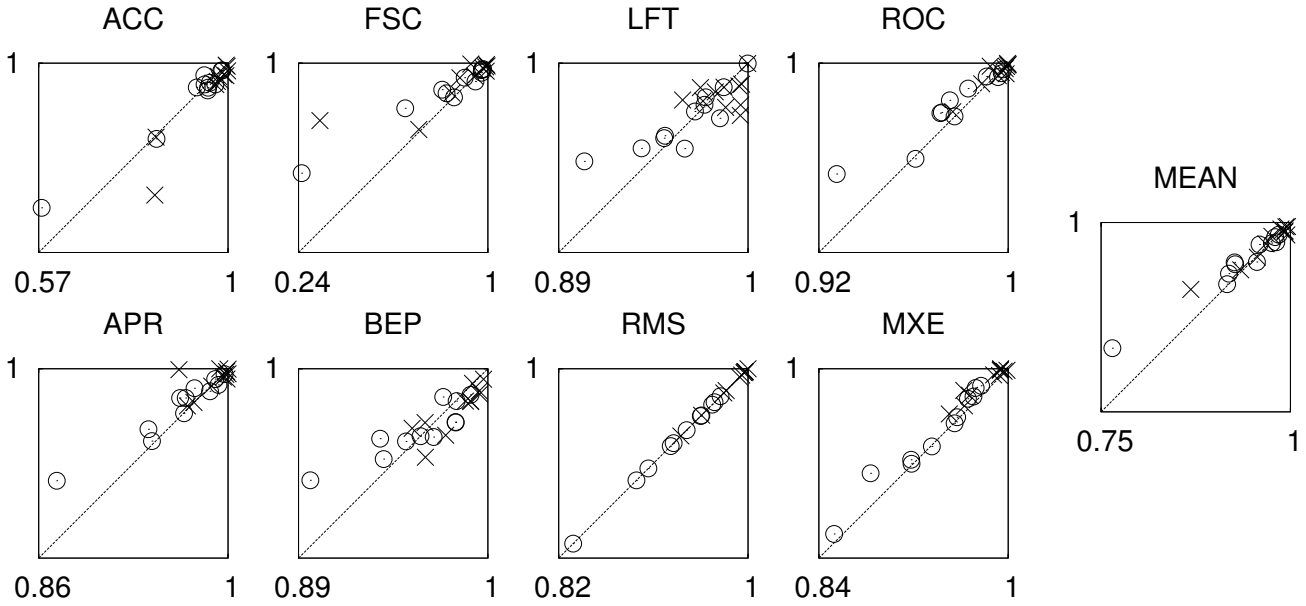

Figure 2. Scatter plots of ensemble selection performance when RMS is optimized (x-axis) vs when the target metric is optimized (y-axis). Points above the line indicate better performance by optimizing to the target metric (e.g. accuracy) than when optimizing RMS. Each point represents a different data set; circles are averages for a problem over 5 folds, and X's are performances using cross-validation. Each metric (and the mean across metrics) is plotted separately for clarity.

formance when employing varying levels of library pruning. The pruning works as follows: the models are sorted by their performance on the target metric (with respect to the hillclimb set), and only the top $X\%$ of the models are used for ensemble selection. Note that this pruning is different from work on ensemble pruning [12, 22, 23, 26, 13]. This is a *pre-processing* method, while ensemble pruning *post-processes* an existing ensemble.

Figure 3 shows the effect of pruning for each performance metric, averaged across the 11 data sets and 5 folds using non-cross-validated ensemble selection with and without bagging. For comparison, flat lines illustrate the performance achieved by model selection (modsel) and non-pruned ensemble selection (es-both). The legend is shown in the ACC graph.

The figure clearly shows that pruning usually does not hurt ensemble selection performance, and often improves it. For ACC, LFT, and BEP pruned ensemble selection (the line with boxes) seems to yield the same performance as non-pruned ensemble selection . For the other metrics, pruning yields superior performance. Indeed, when using more than 50% of the models performance decreases. Interestingly, library pruning reduces the need for bagging, presumably by reducing the potential for overfitting.[3]

The graphs in Figure 3 show the *average* behavior across our 11 data sets. Ensemble selection's behavior under pruning may in fact vary when each data set is considered individually. Averaging across problems could hide different peak points. Figure 4 shows RMS performance for each of the problems.

Although performance starts to decline at different pruning levels for the different problems, it is clear that larger model libraries increase the risk of overfitting the hillclimb set. Using 100% of the models is never worthwhile. At best, using the full library can match the performance of using only a small subset. In the worst case, ensemble selection overfits. This is particularly evident for the COD data set where model selection outperforms ensemble selection unless pruning is employed.

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 into account model diversity (see for example [11, 17]) might find better pruned sets.

[3]The *bagging* line at 100% does not always match the *es-both* line, even though these should be equivalent configurations. This is particularly evident for FSC, the highest variance metric. The sorting performed before pruning alters ensemble selection's model sampling, resulting in additional variance.

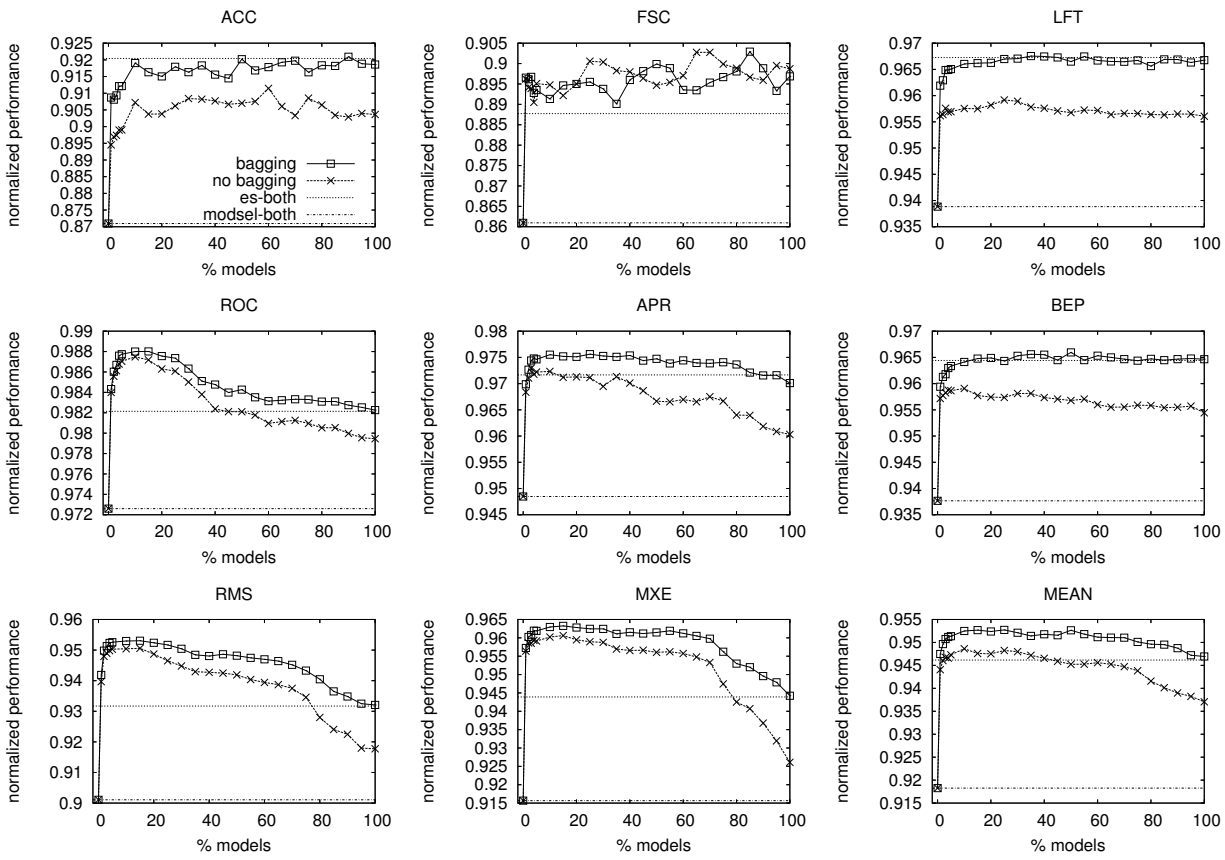

Figure 3. Pruned ensemble selection performance.

9 Discussion

In this section we further analyze the benefit of embedding cross validation within ensemble selection and also briefly describe other work we are doing to make ensemble selection models smaller and faster.

9.1 Benefits of Cross-Validation

The results in Section 6 show that embedding cross-validation within ensemble selection significantly increases the performance of ensemble selection. There are two factors that could explain this increase in performance. First, the bigger hillclimbing set could make selecting models to add to the ensemble more reliable and thus make overfitting harder. Second, averaging the predictions of the sibling models could provide a bagging-like effect that improves the performance of the base-level models. To tease apart the benefit due to each of these factors we perform two additional experiments.

In one experiment, we use the same hillclimbing set as cross-validated ensemble selection, but instead of averaging the predictions of the sibling models, we use only the predictions of *one* of the siblings. Using this procedure we construct five ensemble models, one for each fold, and report their mean performance. This provides a measure of the benefit due to the increase in the size of the hillclimb set (from cross-validation) while eliminating the bagging-like effect due to sibling model averaging.

In the other experiment, we use the smaller hillclimb sets used by *un-cross-validated* ensemble selection, but we do average the predictions of the sibling models. We again construct five ensemble models, one for each fold, and report their mean performance. This allows us to identify the performance increase due to the bagging-like effect of averaging the predictions of the sibling models.

Table 5 shows the results of these experiments. Entries in the table show the improvement provided by using a larger hillclimb set (ES-HILL) and by averaging the sibling models (ES-AVG) as a percentage of the total benefit of cross-validated ensemble selection. For example, looking at the ACC column, increasing the size of the hillclimb set from 1k to 5k yields a benefit equal to 32.9% of the total benefit provided by cross-validated ensemble selection, and aver-

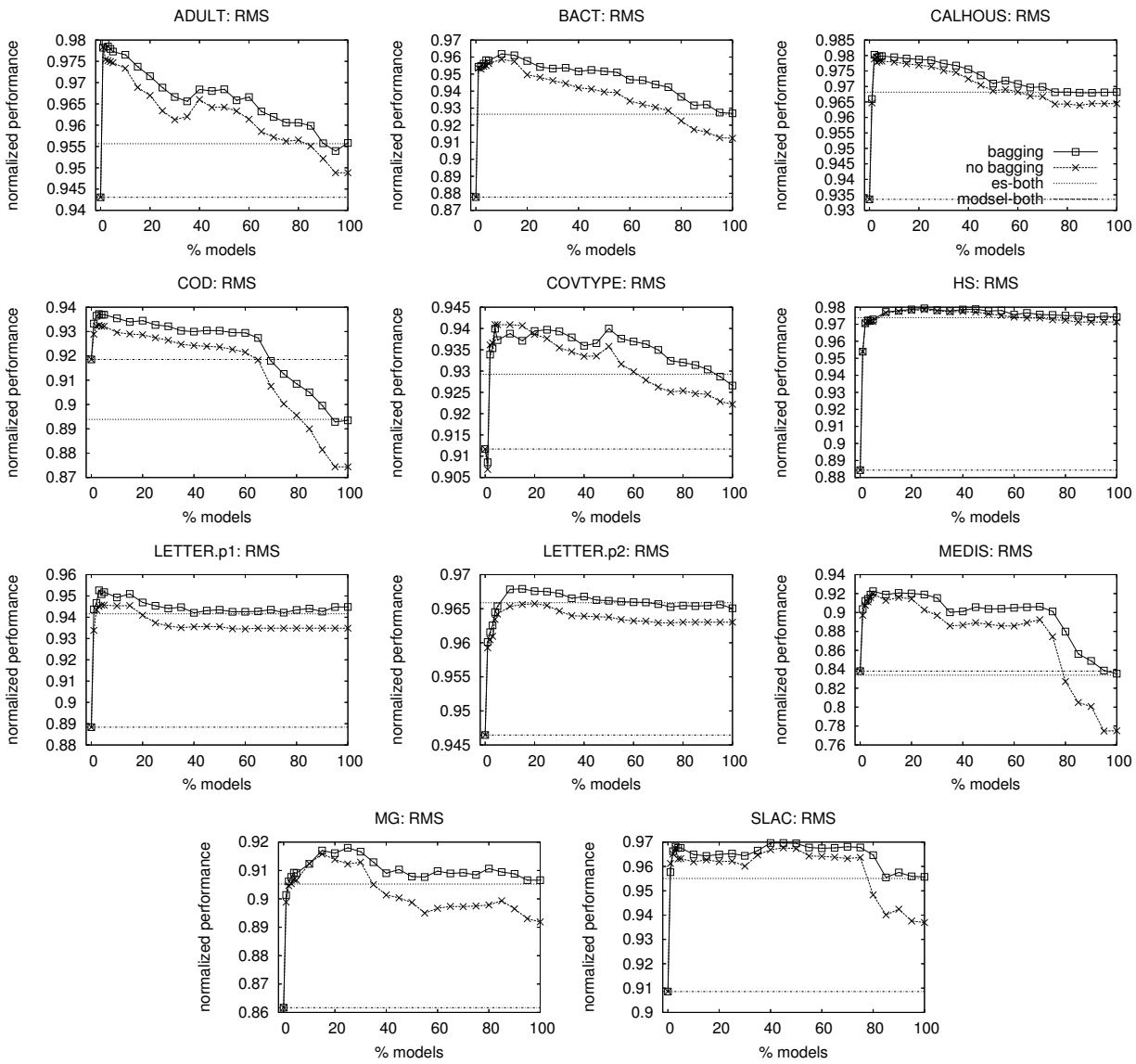

Figure 4. RMS performance for pruned ensemble selection.

aging the sibling models yields a benefit equal to 80.5%.

The third row in the table is the sum of the first two rows. If the sum is lower than 100% the effects from ES-HILL and ES-AVG are super-additive, i.e. combining the two effects provides more benefit than the sum of the individual improvements. If the sum is higher than 100% then the two effects are sub-additive. For ACC, the sum is 113.4%, indicating that the effects of these two factors are sub-additive: the total performance is slightly less than would be expected if the factors were independent. Except for the high variance metrics, FSC and ACC, the sums are close to 100%, indicating that the two effects are nearly independent.

The learning curves in Figure 1 suggest that increasing the size of the hillclimb set from 1k to 5k would explain almost all of the benefit of cross-validation. These results, however, show that on average across the eight metrics the benefit from ES-HILL and ES-AVG are roughly equal. About half of the benefit from embedding cross-validation within ensemble selection appears to result from the increase in the size of the hillclimb set, and the other half appears to result from averaging the sibling models. Increasing the size of the hillclimb set *via cross-validation* (as opposed to having more data available for hillclimbing) provides less benefit in practice because there is a mismatch between the base-level models used to make predictions on the hillclimbing set and the sibling-averaged models that

Table 5. Breakdown of improvement from cross-validation.

	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN
ES-HILL	32.9%	37.2%	48.0%	38.8%	40.8%	19.4%	55.1%	56.7%	41.1%
ES-AVG	80.5%	13.6%	54.0%	59.0%	55.7%	77.4%	46.8%	51.8%	54.9%
SUM	113.4%	50.8%	102.0%	97.8%	96.5%	96.8%	101.9%	108.5%	96.0%

will be used in the ensemble. In other words ensemble selection is hillclimbing using slightly different models than the ones it actually adds to the ensemble.

9.2 Model Compression

While very accurate, the ensembles built by ensemble selection are exceptionally complex. On average, storing the learned ensemble requires 550 MB, and classifying a single test case takes about 0.5 seconds. This prohibits their use in applications where storage space is at a premium (e.g. PDAs), where test sets are large (e.g. Google), or where computational power is limited (e.g. hearing aids). In a separate paper we address these issues by using a *model compression* [6] method to obtain models that perform as well as the ensembles built by ensemble selection, but which are faster and more compact.

The main idea behind model compression is to train a fast and compact model to approximate the function learned by a slow, large, but high performing model. Unlike the true function that is unknown, the function learned by the high performing model is available and can be used to label large amounts of synthetic data. A fast, compact and expressive model trained on enough synthetic data will not overfit and will closely approximate the function learned by the original model. This allows a slow, complex model such as a massive ensemble to be compressed into a fast, compact model with little loss in performance.

In the model compression paper, we use neural networks to compress ensembles produced by ensemble selection. On average the compressed models retain more than 90% of the improvement provided by ensemble selection (over model selection), while being more than 1000 times smaller and 1000 times faster.

10 Conclusions

Embedding cross-validation inside ensemble selection greatly increases its performance. Half of this benefit is due to having more data for hillclimbing; the other half is due to a bagging effect that results from the way cross-validation is embedded within ensemble selection. Unsurprisingly, reducing the amount of hillclimbing data hurts performance because ensemble selection can overfit this data more easily. In comparison to model selection, however, ensemble selection seems much more resistant to overfitting when data is scarce. Further experiments varying the amount of *training data provided to the base-level models* are needed to see if ensemble selection is truly able to outperform model selection by such a significant amount on small data sets.

Counter to our and others' intuition [9], calibrating models to put all predictions on the same scale before averaging them did not improve ensemble selection's effectiveness. Most of calibration's improvement comes from the superior base-level models.

Our experiments show that directly optimizing to a target metric is better than always optimizing to some predetermined metric. That said, always optimizing to RMS or MXE was surprisingly competitive. These metrics may be good optimization proxies if the target metric is too expensive to compute repeatedly during hillclimbing.

Finally, pruning the number of available models reduces the risk of overfitting during hillclimbing while also yielding faster ensemble building. In our experiments pruning rarely hurt performance and frequently improved it.

Acknowledgments

We thank Lars Backstrom for help with exploring alternative model calibration methods and the anonymous reviewers for helpful comments on paper drafts. This work was supported by NSF Award 0412930.

References

[1] S. D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *ICML*, pages 37–45, 1998.

[2] L. Breiman. Heuristics of instability in model selection. Technical report, Statistics Department, University of California at Berkeley, 1994.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] R. K. Bryll, R. Gutierrez-Osuna, and F. K. H. Quek. Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302, 2003.

[6] C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression: Making big, slow models practical. In *Proc. of the 12th International Conf. on Knowledge Discovery and Data Mining (KDD'06)*, 2006.
[7] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *ICML*, 2004.
[8] P. Domingos. Bayesian averaging of classifiers and the overfitting problem. In *ICML*, pages 223–230. Morgan Kaufmann, San Francisco, CA, 2000.
[9] R. P. W. Duin. The combining classifier: To train or not to train? In *ICPR (2)*, pages 765–770, 2002.
[10] P. Giudici. *Applied Data Mining*. John Wiley and Sons, New York, 2003.
[11] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.
[12] D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *ICML*, pages 211–218. Morgan Kaufmann, 1997.
[13] G. Martínez-Munoz and A. Suárez. Pruning in ordered bagging ensembles. In *ICML*, pages 609–616, New York, NY, USA, 2006. ACM Press.
[14] A. Munson, C. Cardie, and R. Caruana. Optimizing to arbitrary NLP metrics using ensemble selection. In *HLT-EMNLP*, pages 539–546, 2005.
[15] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *ICML'05*, 2005.
[16] C. Perlich, F. Provost, and J. S. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *J. Mach. Learn. Res.*, 4:211–255, 2003.
[17] A. H. Peterson and T. R. Martinez. Estimating the potential for combining learning models. In *Proc. of the ICML Workshop on Meta-Learning*, pages 68–75, 2005.
[18] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Adv. in Large Margin Classifiers*, 1999.
[19] F. J. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Knowledge Discovery and Data Mining*, pages 43–48, 1997.
[20] F. Roli, G. Giacinto, and G. Vernazza. Methods for designing multiple classifier systems. In *Multiple Classifier Systems*, pages 78–87, 2001.
[21] R. Schapire. The boosting approach to machine learning: An overview. In *In MSRI Workshop on Nonlinear Estimation and Classification*, 2001.
[22] W. N. Street and Y.-H. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *KDD*, pages 377–382, 2001.
[23] G. Tsoumakas, L. Angelis, and I. Vlahavas. Selective fusion of heterogeneous classifiers. *Intelligent Data Analysis*, 9(6):511–525, 2005.
[24] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.
[25] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
[26] Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.

A Learning Methods

In addition to the learning methods used by Caruana et al. [7] (decision trees, bagged trees, boosted trees and stumps, KNN, neural nets, and SVMs), we use three more model types: logistic regression, naïve bayes, and random forests. These are trained as follows:

Logistic Regression (LOGREG): we train both unregularized and regularized models, varying the ridge parameter by factors of 10 from 10^{-8} to 10^{4}. Attributes are scaled to mean 0 and standard deviation 1.

Random Forests (RF): we use the Weka implementation [24]. The forests have 1024 trees, and the size of the feature set to consider at each split is 1, 2, 4, 6, 8, 12, 16 or 20.

Naïve Bayes (NB): we use the Weka implementation and try all three of the Weka options for handling continuous attributes: modeling them as a single normal, modeling them with kernel estimation, or discretizing them using supervised discretization.

In total, around 2,500 models are trained for each data set. When calibrated models are included for ensemble selection the number doubles to 5,000.

B Data Sets

We experiment with 11 binary classification problems. ADULT, COV_TYPE, HS, LETTER.P1, LETTER.P2, MEDIS, and SLAC were used by Caruana et al. [7]. The four new data sets we use are BACT, COD, CALHOUS, and MG. COD, BACT, and CALHOUS are three of the datasets used in Perlich et al. [16]. MG is a medical data set. See Table 6 for characteristics of the 11 problems.

Table 6. Description of problems

PROBLEM	#ATTR	TRAIN	TEST	%POZ
ADULT	14/104	4000	35222	25%
BACT	11/170	4000	34262	69%
COD	15/60	4000	14000	50%
CALHOUS	9	4000	14640	52%
COV_TYPE	54	4000	25000	36%
HS	200	4000	4366	24%
LETTER.P1	16	4000	14000	3%
LETTER.P2	16	4000	14000	53%
MEDIS	63	4000	8199	11%
MG	124	4000	12807	17%
SLAC	59	4000	25000	50%

Table 7. Scales used to compute normalized scores. Each entry shows bottom / top for the scale.

	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE
ADULT	0.752 / 0.859	0.398 / 0.705	1.000 / 2.842	0.500 / 0.915	0.248 / 0.808	0.248 / 0.708	0.432 / 0.312	0.808 / 0.442
BACT	0.692 / 0.780	0.818 / 0.855	1.000 / 1.345	0.500 / 0.794	0.692 / 0.891	0.692 / 0.824	0.462 / 0.398	0.891 / 0.697
CALHOUS	0.517 / 0.889	0.681 / 0.893	1.000 / 1.941	0.500 / 0.959	0.517 / 0.964	0.517 / 0.895	0.500 / 0.283	0.999 / 0.380
COD	0.501 / 0.784	0.666 / 0.796	1.000 / 1.808	0.500 / 0.866	0.499 / 0.864	0.499 / 0.782	0.500 / 0.387	1.000 / 0.663
COVTYPE	0.639 / 0.859	0.531 / 0.804	1.000 / 2.487	0.500 / 0.926	0.362 / 0.879	0.361 / 0.805	0.480 / 0.320	0.944 / 0.478
HS	0.759 / 0.949	0.389 / 0.894	1.000 / 3.656	0.500 / 0.985	0.243 / 0.962	0.241 / 0.898	0.428 / 0.198	0.797 / 0.195
LETTER.p1	0.965 / 0.994	0.067 / 0.917	1.000 / 4.001	0.500 / 0.999	0.036 / 0.975	0.035 / 0.917	0.184 / 0.067	0.219 / 0.025
LETTER.p2	0.533 / 0.968	0.696 / 0.970	1.000 / 1.887	0.500 / 0.996	0.534 / 0.997	0.533 / 0.970	0.499 / 0.157	0.997 / 0.125
MEDIS	0.893 / 0.905	0.193 / 0.447	1.000 / 2.917	0.500 / 0.853	0.108 / 0.462	0.107 / 0.469	0.309 / 0.272	0.491 / 0.365
MG	0.831 / 0.900	0.290 / 0.663	1.000 / 3.210	0.500 / 0.911	0.170 / 0.740	0.169 / 0.686	0.375 / 0.278	0.656 / 0.373
SLAC	0.501 / 0.726	0.667 / 0.751	1.000 / 1.727	0.500 / 0.813	0.501 / 0.816	0.501 / 0.727	0.500 / 0.420	1.000 / 0.755

C Performance Scales

Table 7 lists the performance numbers that determine the normalized scores. Each entry contains the baseline performance (bottom of the scale) and the best performance achieved by any model or ensemble (top of the scale).