Corrected version of the paper that appeared in the Proceedings of ICML'04.

Ensemble Selection from Libraries of Models

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Geoff Crew GC97@CS.CORNELL.EDU
Alex Ksikes AK107@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

We present a method for constructing ensembles from libraries of thousands of models. Model libraries are generated using different learning algorithms and parameter settings. Forward stepwise selection is used to add to the ensemble the models that maximize its performance. Ensemble selection allows ensembles to be optimized to performance metric such as accuracy, cross entropy, mean precision, or ROC Area. Experiments with seven test problems and ten metrics demonstrate the benefit of ensemble selection.

1. Introduction

An ensemble is a collection of models whose predictions are combined by weighted averaging or voting. Dietterich (2000) states that "A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse."

Many methods have been proposed to generate accurate, yet diverse, sets of models. Bagging (Breiman, 1996) trains models of one type (e.g., C4 decision trees) on bootstrap samples of the training set. Opitz (1999) bags features instead of training points. Boosting (Schapire, 2001) generates a potentially more diverse set of models than bagging by weighting the training set to force new models attend to those points that are difficult to classify correctly. Sullivan et al. (2000) boost features instead of training points. Error-correcting-output-codes (ECOC) (Dietterich & Bakiri, 1995) creates models with decorrelated errors by training models for multi-class problems on different dichotomies. Munro and Parmanto (1996) created diverse neural nets via competition among nodes.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

Here we generate diverse sets of models by using many different algorithms. We use Support Vector Machines (SVMs), artificial neural nets (ANNs), memory-based learning (KNN), decision trees (DT), bagged decision trees (BAG-DT), boosted decision trees (BST-DT), and boosted stumps (BST-STMP). For each algorithm we train models using many different parameter settings. For example, we train 121 SVMs by varying the margin parameter C, the kernel, and the kernel parameters (e.g. varying gamma with RBF kernels.)

We train about 2000 models for each problem. Some models have excellent performance, equal to or better than the best models reported in the literature. Other models, however, have mediocre or even poor performance. Rather than combine good and bad models in an ensemble, we use forward stepwise selection from the library of models to find a subset of models that when averaged together yield excellent performance. The basic ensemble selection procedure is very simple:

1. Start with the empty ensemble.
2. Add to the ensemble the model in the library that maximizes the ensemble's performance to the error metric on a hillclimb (validation) set.
3. Repeat Step 2 for a fixed number of iterations or until all the models have been used.
4. Return the ensemble from the nested set of ensembles that has maximum performance on the hillclimb (validation) set.

Models are added to an ensemble by averaging their predictions with the models already in the ensemble. This makes adding a model to the ensemble *very* fast, allowing ensembles with excellent performance to be found in minutes from libraries with 2000 models. Moreover, the selection procedure allows us to optimize the ensemble to any easily computed performance metric. We evaluate the performance of ensemble selection on ten performance metrics. We believe this is the first time a learning method has been evaluated across such a wide variety of performance metrics.

On each performance metric we compare ensemble selection to the model in the library that performs best on that metric. Because we generate so many different models, libraries usually contain a few models with excellent performance on any performance metric. Just selecting the best single model from a library yields remarkably good performance. Ensemble selection, however, finds ensembles that outperform the best single models. This suggests that using different learning methods and parameter settings generates libraries containing a diverse set of good-performing models.

The parameters we vary for each algorithm to generate 2000 models are described in the Appendix. Note that we do not determine what parameters yield best performance when training models. All models are added to a library no matter how good or bad they are. Model predictions on the train and hillclimbing sets are cached. This simplifies working with the library and makes model selection faster because the models do not have to be executed during selection.

2. Improving Ensemble Selection

The simple forward model selection procedure presented in the Introduction is fast and effective, but sometimes overfits to the hillclimbing set, reducing ensemble performance. We made three additions to this selection procedure to reduce overfitting. These are discussed in the next three sub-sections. These methods may be useful in other applications where forward stepwise selection is prone to overfitting, such as in feature selection (Kohavi & John, 1997).

2.1. Selection with Replacement

With model selection *without* replacement, performance improves as the best models are added to the ensemble, peaks, and then quickly declines. Performance drops because the best models in the library have been used and selection must now add models that hurt the ensemble. Figure 1 shows this behavior for root-mean-squared-error. Unfortunately, most error metrics yield much bumpier graphs than this when hillclimbing is done with small data sets, making it difficult to reliably pick a good stopping point. The loss in performance can be significant if the peak is missed.

Figure 1 also shows that selecting models *with replacement* greatly reduces this problem. Selection with replacement allows models to be added to the ensemble multiple times. Once peak performance is reached, if the unused models all hurt ensemble performance, selection adds models that were added before rather than hurt performance. This flattens the performance curve past the peak, and allows selection to fine tune ensembles by weighting models: models added to the ensemble multiple times receive more weight.

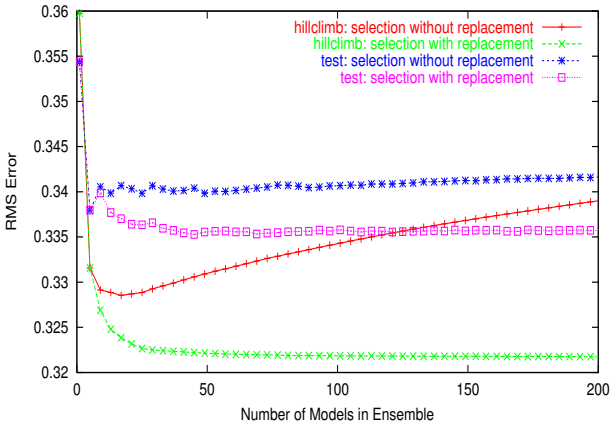

Figure 1. Selection With and Without Replacement.

Selection with replacement flattens the curve so much that a test set is not needed to determine when to stop adding models to the ensemble. The hillclimbing set can be used to stop hillclimbing. This means ensemble selection does not need more test sets than the base-level models would have used to select model parameters. Ensemble selection uses the validation set to do both parameter *and* model selection.

2.2. Sorted Ensemble Initialization

Forward selection sometimes overfits early in selection when ensembles are small. One way to prevent this is to initialize ensembles with more models. Instead of starting with an empty ensemble, sort the models in the library by their performance, and put the best N models in the ensemble. N is chosen by looking at performance on the hillclimbing set. This typically adds 5-25 of the best models to an ensemble *before* greedy stepwise selection begins. Since each of the N best models performs well, they form a strong initial ensemble and it is more difficult for greedy selection to find models that overfit when added to the ensemble.

2.3. Bagged Ensemble Selection

As the number of models in a library increases, the chances of finding combinations of models that overfit the hillclimbing set increases. Bagging can minimize this problem. We reduce the number of models selection can choose from by drawing a random sample of models from the library and selecting from that sample. If a particular combination of M models overfits, the probability of those M models being in a random bag of models is less than $(1-p)^M$ for p the fraction

of models in the bag. We use $p = 0.5$, and bag ensemble selection 20 times to insure that the best models will have many opportunities to be selected. The final ensemble is the average of the 20 ensembles. Bags of ensembles seem complex, but each ensemble is just a weighted average of models, so the average of a set of ensembles also is a simple weighted average of the base-level models. Bagging is discussed in Section 5.3.

3. Data Sets

We experiment with seven problems: ADULT, COVER_TYPE, LETTER.p1, and LETTER.p2 from the UCI Repository (Blake & Merz, 1998), MEDIS, a pneumonia data set, SLAC, data from collaborators at the Stanford Linear Accelerator, and HYPER_SPECT, the Indian Pines hyperspectral data (Gualtieri et al., 1999). ADULT, MEDIS and SLAC are binary problems. COVER_TYPE, LETTER, and HYPER_SPECT are 7, 26, and 16 class problems, respectively. We converted these to binary because many of the metrics we study are defined on binary problems, and because learning methods such as SVMs and boosting are easier to apply to binary problems. COVER_TYPE was converted to binary by treating the largest class as class 1. LETTER was converted two ways. LETTER.p1 treats the confusable letter 'O' as class 1 and the remaining 25 letters as class 0, yielding a very unbalanced binary problem. LETTER.p2 uses letters 1-13 as class 0 and letters 14-26 as class 1, yielding a difficult, but balanced, problem. HYPER_SPECT was converted to binary by treating the large confusable class Soybean-Mintil as class 1.

These data sets were selected because they are large enough to allow moderate size train and validation sets, and still have data left for large final test sets. For our experiments, we used training sets of 5000 points. Each training sample was split into a train set of 4000 points and a hillclimbing/validation set of 1000 points. The final test sets for most of these problems contain 20,000 points – enough to make discerning small differences in performance reliable.

4. Performance Metrics

We use ten performance metrics: accuracy (ACC), root-mean-squared-error (RMS), mean cross-entropy (MXE), lift (LFT), precision/recall break-even point (BEP), precision/recall F-score (FSC), average precision (APR), ROC Area (ROC), and a measure of probability calibration (CAL). The tenth metric is $SAR = (ACC + ROC + (1 - RMS))/3$. SAR is a robust metric to use when the correct metric is unknown. An attractive feature of ensemble selection is that it *can* optimize to metrics such as SAR. We compare performance using ten metrics because different metrics are appropriate in different settings and because learning methods that perform well on one metric do not always perform well on other metrics.

5. Empirical Results

In this section we compare ensemble selection to the best models trained with any of the learning algorithms, and also to several other ensemble methods.

5.1. Normalized Scores

Performance metrics such as ACC or RMS range from 0 to 1, while others (LFT, MXE) range from 0 to p where p depends on the data. For some metrics lower values indicate better performance. For others higher values are better. The baseline rates of metrics such as ROC are independent of the data, while others such as ACC have baseline rates that depend on the data. If baseline ACC = 0.98, ACC = 0.981 probably is poor performance, but if the Bayes optimal ACC = 0.60, achieving ACC = 0.59 might be excellent performance.

To allow averaging across problems and metrics, we convert performances to a normalized $[0, 1]$ scale. On this scale, 0 represents baseline performance for the metric and problem (e.g., 0.5 for ROC, or the baseline ACC for each problem), and 1 represents the best performance seen for any model on the final test set.[1] CAL, the metric used to measure probability calibration, is unusual in that the baseline model that predicts p for all cases, where p is the percent of positives in the test set, has excellent calibration.[2] This creates a problem when normalizing CAL scores because the baseline model and Bayes optimal model have similar CAL scores. Unlike the other measures, CAL is scaled so that the minimum observed CAL score is 0.0 and the mean observed CAL score is 1.0.

The bottom of Table 1 shows normalized scores for each learning algorithm when its parameters are selected for each problem and metric using the validation sets ensemble selection hillclimbs on. Entries in the table are averages over the seven problems. Scores

[1] We would set the top of the scale to the Bayes optimal performance for each problem and metric if we knew it.

[2] Because of this, measures like CAL typically are not used alone, but are used in conjunction with other measures such as ROC to insure that only models with good discrimination *and* good calibration are selected. This does not mean CAL is a poor metric – it is effective at distinguishing poorly calibrated models from well calibrated models.

Table 1. Normalized Scores for the Best Single Models of Each Type (bottom of tbl), and for Ensemble Selection, Bayesian Model Averaging, Stacking with Regression, Averaging All Models, and Picking the Best Model of Any Type (top of tbl).

MODEL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	CAL	SAR	MEAN
ENS. SEL.	**0.956**	**0.944**	**0.992**	**0.997**	**0.985**	**0.979**	**0.980**	**0.981**	0.906	**0.996**	**0.969**
BAYESAVG	0.926	0.891	0.979	0.985	0.977	0.956	0.950	0.959	0.907	0.941	0.948
BEST	0.928	0.919	0.975	0.988	0.959	0.958	0.919	0.944	**0.924**	0.924	0.946
AVG_ALL	0.836	0.801	0.982	0.988	0.972	0.961	0.827	0.809	0.832	0.916	0.890
STACK_LR	0.275	0.777	0.835	0.799	0.786	0.847	0.332	-0.990	-0.011	0.705	0.406
SVM	0.813	**0.909**	0.948	0.962	0.933	0.938	**0.877**	0.878	0.889	**0.905**	**0.905**
ANN	0.877	0.875	0.949	0.955	0.917	0.914	0.853	0.863	**0.916**	0.896	0.902
BAG-DT	0.811	0.861	0.947	0.967	0.942	0.922	0.859	**0.894**	0.786	0.904	0.888
KNN	0.756	0.846	0.909	0.937	0.885	0.889	0.761	0.735	0.876	0.847	0.844
BST-DT	**0.890**	0.899	**0.957**	**0.978**	**0.960**	**0.943**	0.607	0.611	0.413	0.871	0.806
DT	0.526	0.789	0.850	0.868	0.767	0.795	0.556	0.624	0.720	0.745	0.722
BST-STMP	0.732	0.790	0.906	0.919	0.861	0.834	0.304	0.286	0.389	0.659	0.669

near 1 indicate that a model performs close to the best performance observed on all seven problems. Negative entries mean the best models perform below baseline. Bold entries in the bottom of Table 1 show which learning algorithms yield the best performance on each metric. The last column is the mean normalized score across the ten metrics.

Ignoring the ensemble methods in the top of the table, the best algorithms overall are SVMs, ANNs, and bagged trees, with mean normalized scores 0.905, 0.902, and 0.888, respectively. Boosted trees do not perform well on the probability metrics (RMS, MXE, and CAL), but are excellent on the threshold metrics (ACC, FSC, and LFT) and ordering metrics (ROC, APR, and BEP). If mean performance was measured over just these six metrics, boosted trees would outperform bagged trees, SVMs, and ANNs.

The top of Table 1 shows normalized scores for ensemble selection, Bayesian model averaging, the best individual models of any type (BEST), a simple average of all models (AVG_ALL), and stacking with logistic regression (STACK_LR). Stacking (Wolpert, 1992) with logistic regression performs poorly because regression overfits dramatically when there are 2000 highly correlated input models and only 1k points in the validation set. An unweighted average of all the models (AVG_ALL) is better than weighting the models via regression because unweighted averaging cannot overfit the validation set. Averaging, however, performs worse than just picking the best single models because some of the models have poor performance and thus hurt the ensembles. As expected, picking the best single model from the library (BEST) performs better than any one learning algorithm in the bottom of the table because it can pick the best model from any learning method for each metric and test problem.

Bayesian model averaging (Domingos, 2000) has a mean score of 0.948, significantly better than SVMs (the best single model). In Bayesian model averaging, each model in the library is weighted by the likelihood of the data (validation set) given the model when model predictions are treated as probabilities:

$$W_M = \prod_{i=1}^{N} Targ_i * Pred_i + (1 - Targ_i) * (1 - Pred_i)$$

where W_M is the weight of model M, $Targ_i$ is the 0/1 target for case i, and $Pred_i$ is the predicted probability that case i is class 1. Note that Bayesian model averaging outperforms BEST on only 5 of the 10 metrics.

With a mean normalized score of 0.969, ensemble selection is the clear winner. It outperforms the other ensemble methods (and the best individual models) on all 10 of the 10 metrics (significant at $p = 0.001$). We believe ensemble selection consistently outperforms the other ensemble methods for two reasons:

- ensemble selection is able to optimize the ensemble differently for each performance metric
- overfitting is a serious problem when there are 2000 models to combine. The methods presented in Section 2 to combat overfitting contribute to ensemble selection's excellent performance.

The consistently strong performance of ensemble selection suggests that using many different learning methods and parameter settings is an effective way of generating a diverse collection of models. The consistent performance also suggests that forward stepwise selection is an effective way of selecting high-performance

ensembles from these models for a variety of performance metrics if overfitting is carefully controlled.

5.2. Percent Reduction in Loss (Error)

In this section we compare ensemble selection to the best single models (BEST in Table 1) using percent reduction in error. One advantage of percent reduction in error compared to normalized scores is that normalized scores change as better models are found and shift the top of the scale up. The percent reduction in error from model A to B depends only on the performances of models A and B, not on some other model that defines the top of the performance scale.

To calculate percent reduction in error, we first convert each metric to loss where 0 represents perfect performance and 1 represents worst performance. Perfect prediction yields $ACC = BEP = FSC = ROC = APR = SAR = 1$ and $RMSE = MXE = CAL = 0$, all of which have loss 0. An example will help. If ACC improves from 0.75 to 0.77, the losses are 0.25 and 0.23, respectively, and the percent reduction in error is 8%. One potential disadvantage of percent reduction in loss is that it gives more emphasis to reductions at low loss: reducing loss from 0.02 to 0.01 is a 50% reduction, wheras reducing loss from 0.20 to 0.19 (same absolute change), is only a 5% reduction. In some domains this bias is appropriate. In others it is not. Normalized scores do not have this bias.

Table 2 shows the percent reduction in loss for ensemble selection on the 7 test problems and 10 metrics, compared to the best models selected for each problem and metric. As with normalized scores, final performances are estimated on large final test sets not used for training and is the average over two trials with each problem. Positive entries in the table mean error was reduced, i.e. ensemble selection performed better.

Ensemble selection wins 64 of 70 times (significant at $p = 0.001$). On average, ensemble selection reduces loss 8.76% over the best models for each problem and metric. To make this concrete, if the best model has accuracy 90.00%, loss is 0.1000, and an 8.7% reduction in loss corresponds to reducing loss to 0.0913 or increasing accuracy to 90.87%. Similarly, if the best model has RMS 0.2000, an 8.7% reduction in loss corresponds to reducing RMS to 0.1826. Test sets contain 20,000 cases. Increasing accuracy 0.87% means 174 more cases are predicted correctly by the ensembles.

The improvement is not dramatic, but consistently increasing accuracy 0.87% or reducing RMS 0.0174 compared to the best of 2000 models is impressive when the best models have such good performance. But we don't want to overstate the improvement of ensemble selection. For comparison, bagging or boosting trees yields a 20% reduction in loss compared to raw trees, 2.5 times the benefit of ensemble selection. But then, improving the performance of mediocre, high variance models such as decision trees is easier than improving the performance of the very best models.

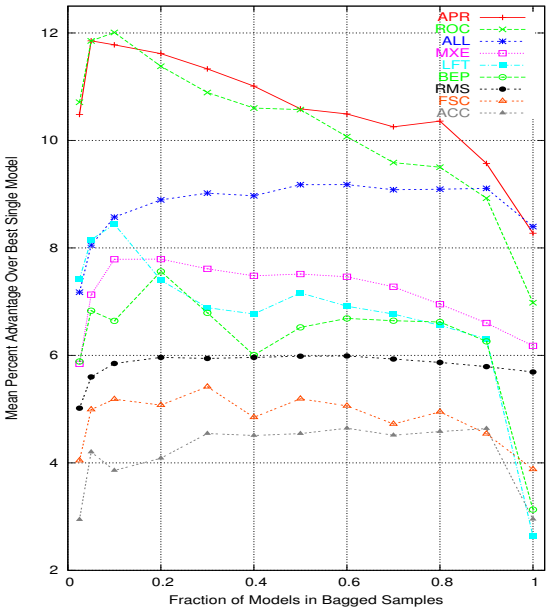

Figure 2. Benefit of Bagged Selection on the 10 Metrics. The right side of the graph at $p = 1$ represents no bagging.

5.3. Bagging to Minimize Overfitting

Figure 2 shows the percent reduction in loss of ensemble selection with bagging as p, the fraction of models in the bags, varies from 1 down to 0.025 for the 10 metrics averaged across the 7 problems. At $p = 1$, bags contain all models which is equivalent to not bagging. On average, bagged ensemble selection reduces loss an additional 2.5% at $p = 0.5$. The results in Tables 1 and 2 are for $p = 0.5$, a value we selected before looking at these results, and which Figure 2 shows is suboptimal. This 2.5% is about a third of the total 8.7% benefit we see with ensemble selection. Figure 2 (which uses the final test sets) suggests p in the range $0.1 - 0.3$ would yield further improvement. We are currently experimenting with using cross validation to pick a near optimal value of p for each problem and metric.

6. Case Study: Classifying Sub-Atomic Particles

Here we present a case study where optimizing an ensemble to the correct metric has impact on an applica-

Table 2. Percent Reduction in Loss of Ensemble Selection Over the Best Models From Any Learning Algorithm.

PROBLEM	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	CAL	SAR	MEAN
ADULT	3.63	1.98	6.28	5.40	5.82	3.60	0.71	1.36	9.30	4.02	4.21
COVER_TYPE	*-0.42*	*-1.08*	4.63	*-1.27*	3.52	*-0.87*	3.79	6.66	1.41	4.05	2.042
LETTER.P1	17.47	26.72	0.10	41.88	28.75	15.02	24.82	15.34	2.61	34.74	20.745
LETTER.P2	3.24	4.27	100.00	0.96	1.93	10.13	4.46	9.65	*-40.03*	6.27	10.088
MEDIS	0.30	*-6.39*	0.83	2.53	4.00	6.53	0.04	0.12	9.48	1.57	1.901
HYPER_SPECT	5.40	5.29	5.98	20.96	27.28	13.55	7.71	18.75	20.89	11.91	13.772
SLAC	2.19	5.55	8.85	3.56	2.82	2.18	0.36	0.71	5.12	1.67	3.301
MEAN	4.544	5.191	18.096	10.574	10.589	7.163	5.984	7.513	1.254	9.176	8.008

tion of machine learning to a particle physics problem. At the Stanford Linear Accelerator Center (SLAC), high energy particle beams are collided to generate subatomic particles. A major challenge in these experiments is to correctly classify the particle tracks. Performance is measured with the SLAC Q-score:

$$\text{SLQ} = \varepsilon(1-2w)^2$$

where ε is the percent of events accepted for prediction, and w is the probability of misclassification. SLQ is an application-specific performance metric that estimates the statistical power of the model. Increasing SLQ by 5% is equivalent to having 5% more data, which potentially saves hundreds of thousands of dollars or more in accelerator time.

Bagged trees have the best SLQ performance, increasing SLQ 0.0355 (a 12% improvement) over well-optimized neural nets and decision trees. SLQ behaves like a cross between accuracy and calibration, so it is not surprising that bagged trees – the models with the best calibration performance – are best on SLQ. When an ensemble is optimized to SLQ with ensemble selection, SLQ performance increases 6% over the best bagged trees. This 6% increase in effective sample size represents a large potential savings in accelerator time.

7. Discussion and Future Work

7.1. Validation and Hillclimbing Sets

Ensemble selection uses the *validation* set to "train" ensembles. Hillclimbing on the validation set does not give ensemble selection an unfair advantage over other models. The validation set would be needed to select the parameters for each algorithm (parameter selection), and then to pick the best algorithm (model selection). Ensemble selection uses the validation set for parameter selection, model selection, and ensemble creation. With many algorithms validation data can be put back in the train set and the model retrained once parameters are selected. This can also be done with ensemble selection. Any strategy for using and reusing validation sets, including cross validation, can be used with ensemble selection. We currently are running experiments with 5-fold cross validation to increase the size of the hillclimbing set to include all of the training data, not just the held-out 1k samples.

7.2. Models Selected by the Ensembles

Table 3 shows the total weight given to each type of model for each metric on ADULT and COVER_TYPE. The average across the ten metrics (right column) shows that KNN and BST_DT receive the most weight in the ensemble for COVER_TYPE. The weights are strikingly different for ADULT where BST_STMP, ANN, and DT receive the most weight. There also are substantial differences between the model types preferred for different metrics. For example, on COVER_TYPE, FSC gives high weight to boosted trees and low weight to KNN, but these weights are reversed for RMS and LFT. Also, ANNs get modest weight for MXE and RMS (what the ANNs optimize), but low weight for ACC and LFT. This suggests that ensemble selection is able to exploit the different strengths and biases of the different learning algorithms when optimizing an ensemble to each metric and to each problem.

7.3. Computational Cost

Building libraries is expensive. Because models are independent, it is easy to parallelize model creation and distribute training across machines. It takes about 48 hours to train the 2000 models using a cluster of ten Linux machines. Model training is automated. There is no parameter tuning or examining performance on validation sets. Usually no one model is critical, so it is not necessary to wait until all models are trained to use the library. This provides an any-time flavor

Table 3. Aggregate Weight Given to Different Types of Models in the Ensembles.

ADULT	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	SAR	AVG_WT
ANN	0.0709	0.1316	0.1012	0.3646	0.4304	0.2432	0.1667	0.0941	0.5732	0.2720
KNN	0.0205	0.0148	0.5865	0.0368	0.0286	0.0486	0.0000	0.0000	0.0492	0.0981
SVM	0.0006	0.0000	0.1096	0.2841	0.2743	0.0924	0.0574	0.0000	0.0224	0.1051
DT	0.0201	0.0346	0.0070	0.0879	0.0491	0.0188	0.7456	0.8674	0.2345	0.2581
BAG_DT	0.0021	0.0008	0.0000	0.0038	0.0051	0.0022	0.0057	0.0102	0.0136	0.0054
BST_DT	0.1100	0.1523	0.0250	0.0572	0.0319	0.0470	0.0245	0.0283	0.0748	0.0689
BST_STMP	0.7759	0.6658	0.1707	0.1657	0.1806	0.5478	0.0000	0.0000	0.0321	0.3173

COV_TYPE	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	SAR	AVG_WT
ANN	0.0114	0.0102	0.0007	0.0382	0.0228	0.0086	0.0871	0.0966	0.0521	0.0410
KNN	0.1790	0.1665	0.5759	0.2518	0.2948	0.2016	0.4364	0.4266	0.3643	0.3621
SVM	0.0206	0.0161	0.0878	0.1039	0.1058	0.0507	0.0103	0.0134	0.0379	0.0558
DT	0.0606	0.0539	0.0124	0.2381	0.2423	0.0295	0.4083	0.3685	0.2000	0.2017
BAG_DT	0.0049	0.0056	0.0019	0.0092	0.0150	0.0064	0.0156	0.0222	0.0438	0.0156
BST_DT	0.5533	0.6134	0.1298	0.2784	0.2397	0.6438	0.0423	0.0727	0.2923	0.3582
BST_STMP	0.1702	0.1343	0.1914	0.0804	0.0797	0.0593	0.0000	0.0000	0.0095	0.0906

to ensemble selection: ensembles can be trained using whatever models are available when the ensemble is needed. It is easy to add more models later. Libraries can be built before the performance metric is known because the libraries themselves do not depend on the metric that will be used to optimize the ensemble. This makes model libraries very reusable.

Forward stepwise ensemble selection is efficient. Adding a model to an ensemble only requires averaging a model's predictions with the ensemble's predictions, which is $O(D)$ for D the size of the hillclimbing set. If there are M models to choose from, this is done M times for each selection step. If selection is run K steps, the cost of ensemble selection is only O(D*M*K) assuming the metric can be computed in O(D). If the metric is more expensive than O(D) (e.g., ROC requires sorting and thus is O(D*logD)), recomputing the metric dominates. Using a JAVA implementation, selecting an ensemble from a library of $M = 2000$ models, a hillclimbing set with $D = 1000$ points, and $K = 200$ takes about a minute on a medium-power workstation. If selection is bagged 20 times, it takes about 20 minutes to build the final ensemble.

7.4. Optimizing To Any Performance Metric

ANNs usually are trained to minimize cross entropy or squared error. Trees and SVMs usually maximize accuracy. Boosting also is designed to maximize accuracy. Some metrics such as precision/recall and ROC are hard to optimize to. Because model averaging is fast, ensemble selection can try adding every model in the library to the ensemble at each step. If the performance of each of these ensembles can be evaluated quickly on the metric, the ensemble can be optimized to that metric by this greedy, brute force search. A good ensemble usually will be found if some base-level models or combinations of them yield good performance on that metric. Although we do not know how to optimize the base-level models to many of these metrics, the ensemble can be optimized to them.

7.5. Beyond Binary Classification

Ensemble selection is straightforward for binary classification and regression. If the base-level models make predictions for multiple classes, no modification to the ensemble selection procedure is necessary for multi-class problems. If some base-level models make predictions one dichotomy at a time (e.g. SVMs), ensemble selection is easiest if the base-level models are combined so that they return a predicted probability for each class. We have not yet experimented with multi-class ensemble selection.

8. Conclusions

Ensemble selection uses forward stepwise selection from libraries of thousands of models to build ensembles that are optimized to the given performance metric. Using a variety of learning algorithms and parameter settings appears to be effective for generating libraries of diverse, high quality models. Ensemble selection's most important feature is that it can optimize ensemble performance to any easily computed performance metric. Experiments with seven test problems

and ten performance metrics show that ensemble selection consistently finds ensembles that outperform all other models, including models trained with bagging, boosting, and Bayesian model averaging.

9. Appendix: Building Model Libraries

KNN: we use 26 values of K ranging from $K = 1$ to $K = |trainset|$. We use KNN with Euclidean distance and Euclidean distance weighted by gain ratio. We also use distance weighted KNN, and locally weighted averaging. The kernel widths for locally weighted averaging vary from 2^0 to 2^{10} times the minimum distance between any two points in the train set.

ANN: we train nets with gradient descent backprop and vary the number of hidden units {1, 2, 4, 8, 32, 128} and the momentum {0, 0.2, 0.5, 0.9}. We don't use validation sets to do weight decay or early stopping. Instead, we stop the nets at many different epochs so that some nets underfit or overfit.

DT: we vary the splitting criterion, pruning options, and smoothing (Laplacian or Bayesian smoothing). We use all of the DT models in Buntine's IND package: Bayes, ID3, CART, CART0, C4, MML, and SMML. We also generate trees of type C44 (C4 with no pruning), C44BS (C44 with Bayesian smoothing), and MMLLS (MML with Laplacian smoothing). See (Provost & Domingos, 2003) for descriptions of C44.

BAG-DT: we bag 25 trees of each type. Each tree trained on a bootstrap sample is added to the library, as well as the final bagged ensemble that averages all these trees. With **BST-DT** we boost each tree type. Boosting can overfit, so we add boosted DTs to the library after {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048} steps of boosting. With **BST-STMP** we use stumps (single level decision trees) with 5 different splitting criteria, each boosted {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192} steps.

SVMs: we use most kernels in SVMLight (Joachims, 1999) {linear, polynomial degree 2 & 3, radial with width {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2}} and vary the regularization parameter C by factors of ten from 10^{-7} to 10^3. The output range of SVMs is $[-\infty, +\infty]$ instead of $[0, 1]$. To make the SVM predictions compatible with other models, we use Platt's method to convert SVM outputs to probabilities by fitting them to a sigmoid (Platt, 1999).

Acknowledgments

Charles Chiu and Kohsuke Kawaguchi helped with the initial design of ensemble selection. We thank Tony Gualtieri for help with the HYPER_SPECT data, and our collaborators at SLAC for help with the SLAC data and SLQ performance metric.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *First International Workshop on Multiple Classifier Systems*, 1–15.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research, 2.*

Domingos, P. (2000). Bayesian averaging of classifiers and the overfitting problem. *Proc. 17th International Conf. on Machine Learning* (pp. 223–230). Morgan Kaufmann, San Francisco, CA.

Gualtieri, A., Chettri, S. R., Cromp, R., & Johnson, L. (1999). Support vector machine classifiers as applied to aviris data. *Proc. Eighth JPL Airborne Geoscience Workshop.*

Joachims, T. (1999). Making large-scale svm learning practical. *Advances in Kernel Methods.*

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97.*

Meek, C., Thiesson, B., & Heckerman, D. (2002). *Staged mixture modeling and boosting* (Technical Report MSR-TR-2002-45).

Munro, P., & Parmanto, B. (1996). Competition among networks improves committee performance. *Advances in Neural Information Processing Systems.*

Opitz, D. (1999). Feature selection for ensembles. *AAAI/IAAI* (pp. 379–384).

Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers* (pp. 61–74).

Provost, F., & Domingos, P. (2003). Tree induction for probability-based rankings. *Machine Learning, 52.*

Schapire, R. (2001). The boosting approach to machine learning: An overview. *In MSRI Workshop on Nonlinear Estimation and Classification.*

Sullivan, J., Langford, J., Caruana, R., & Blum, A. (2000). Featureboost: A meta-learning algorithm that improves model robustness. *Proceedings of the Seventeenth International Conference on Machine Learning.*

Wolpert, D. H. (1992). Stacked generalizatio. *Neural Networks*, *5*, 241–259.

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;

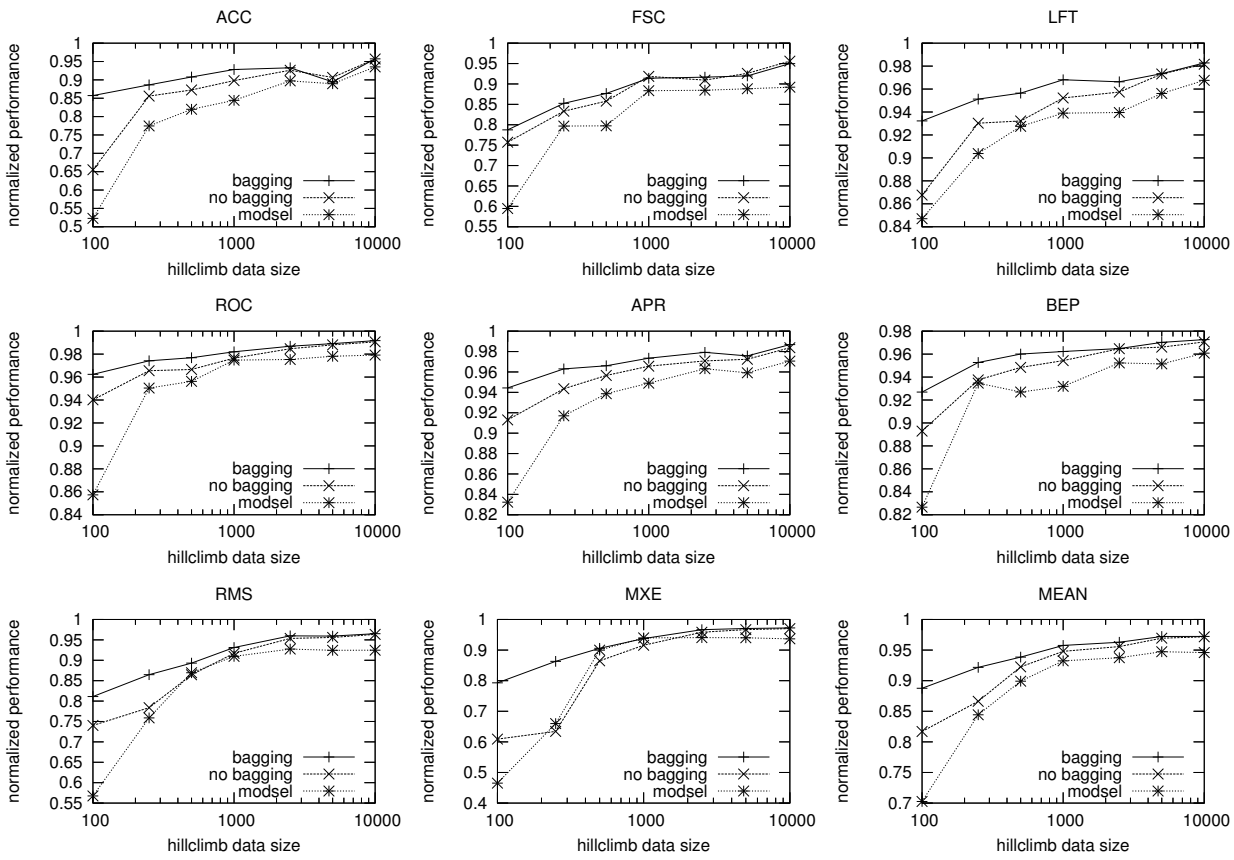

Figure 1. Learning curves for ensemble selection with and without bagging, and for picking the best single model (modsel).

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-

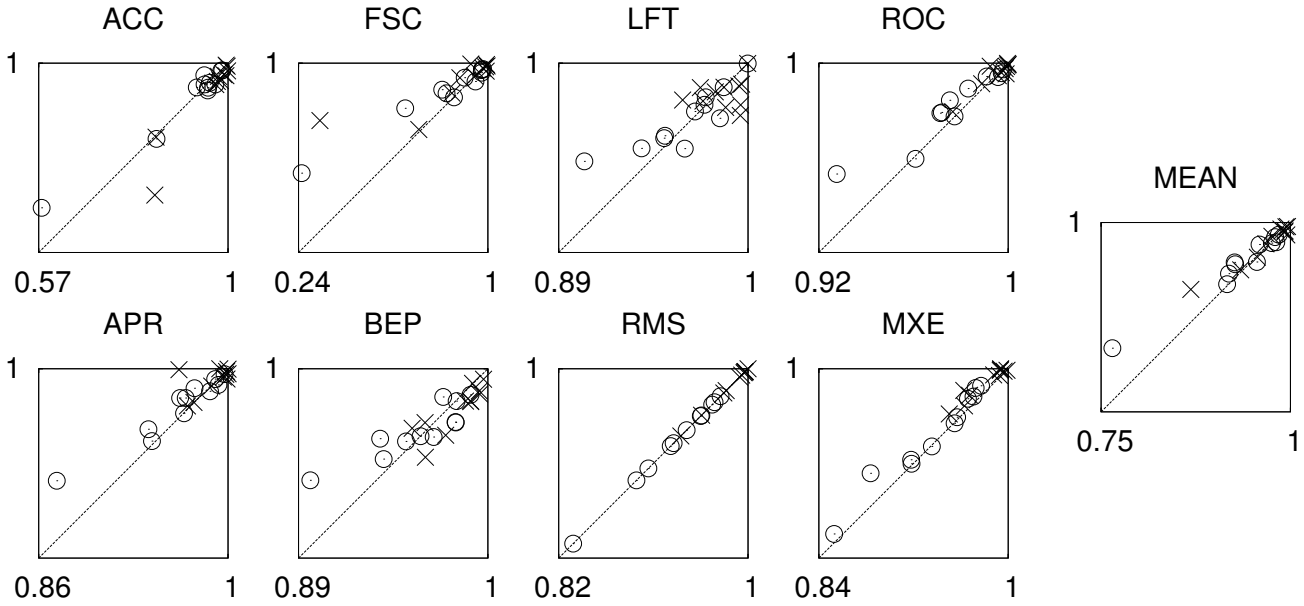

Figure 2. Scatter plots of ensemble selection performance when RMS is optimized (x-axis) vs when the target metric is optimized (y-axis). Points above the line indicate better performance by optimizing to the target metric (e.g. accuracy) than when optimizing RMS. Each point represents a different data 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.

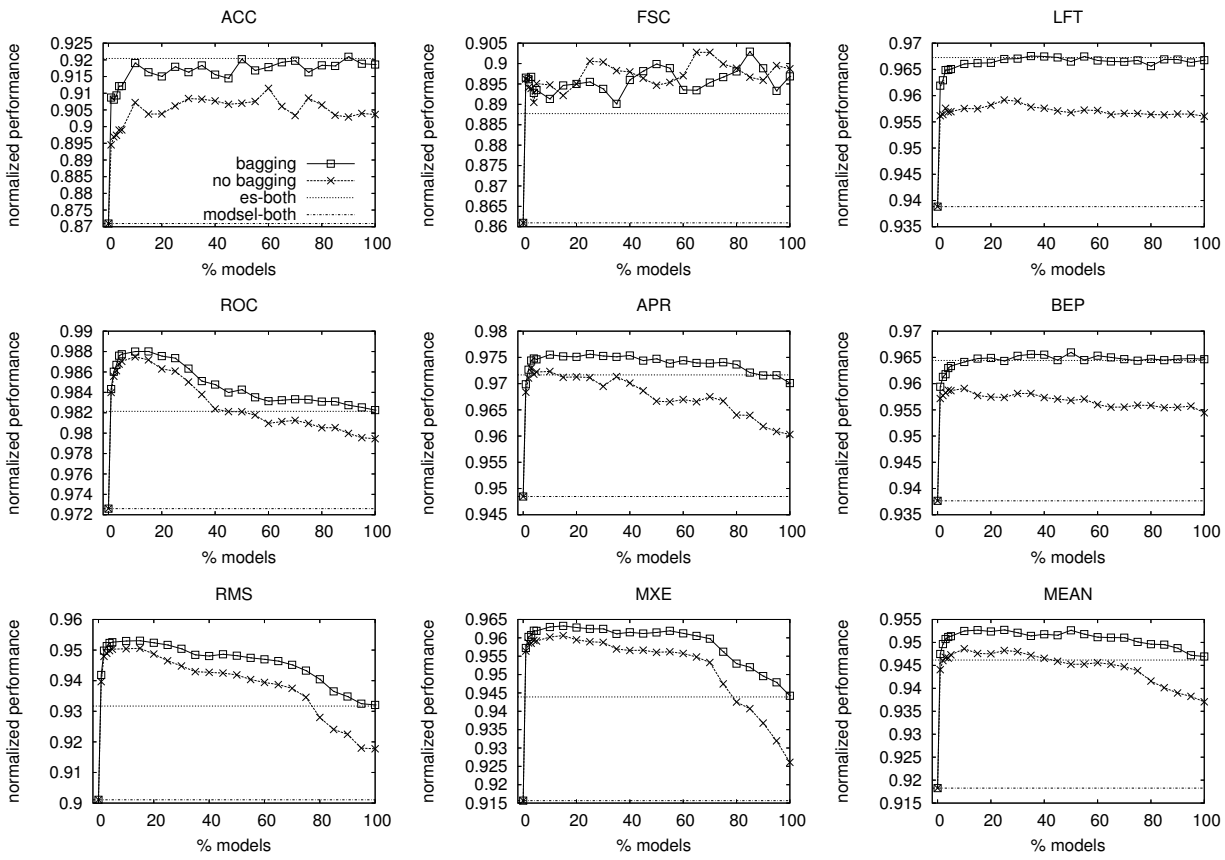

Figure 3. Pruned ensemble selection performance.

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-

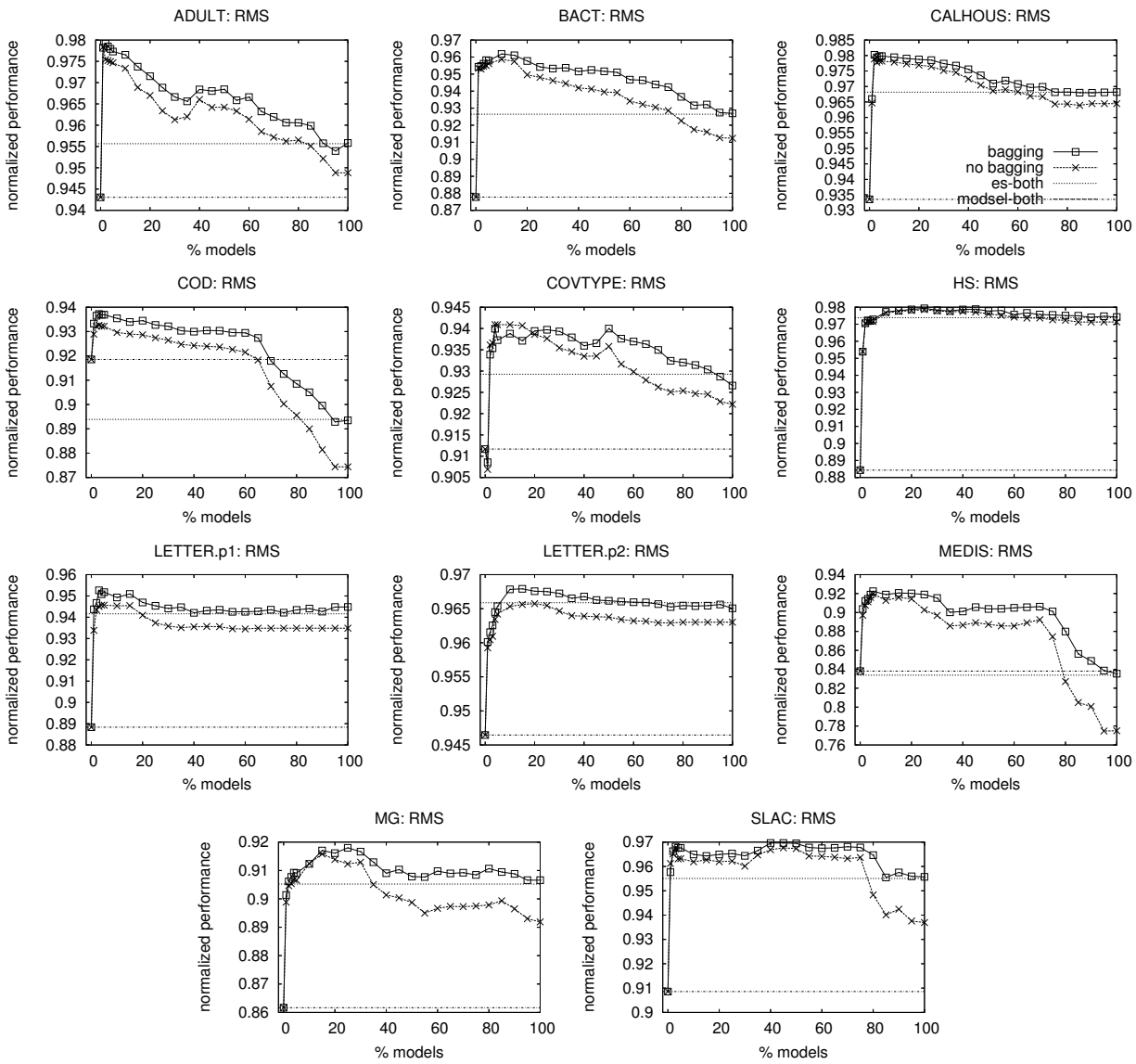

Figure 4. RMS performance for pruned ensemble selection.

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).