Journal of Machine Learning Research 7 (2006) 1315–1338 Submitted 8/05; Revised 4/06; Published 7/06

# Ensemble Pruning Via Semi-definite Programming

**Yi Zhang** YI-ZHANG-2@UIOWA.EDU
**Samuel Burer** SAMUEL-BURER@UIOWA.EDU
**W. Nick Street** NICK-STREET@UIOWA.EDU
*Department of Management Sciences*
*University of Iowa*
*Iowa City, IA 52242-1944, USA*

**Editors:** Kristin P. Bennett and Emilio Parrado-Hernández

## Abstract

An ensemble is a group of learning models that jointly solve a problem. However, the ensembles generated by existing techniques are sometimes unnecessarily large, which can lead to extra memory usage, computational costs, and occasional decreases in effectiveness. The purpose of ensemble pruning is to search for a good subset of ensemble members that performs as well as, or better than, the original ensemble. This subset selection problem is a combinatorial optimization problem and thus finding the exact optimal solution is computationally prohibitive. Various heuristic methods have been developed to obtain an approximate solution. However, most of the existing heuristics use simple greedy search as the optimization method, which lacks either theoretical or empirical quality guarantees. In this paper, the ensemble subset selection problem is formulated as a quadratic integer programming problem. By applying semi-definite programming (SDP) as a solution technique, we are able to get better approximate solutions. Computational experiments show that this SDP-based pruning algorithm outperforms other heuristics in the literature. Its application in a classifier-sharing study also demonstrates the effectiveness of the method.

**Keywords:** ensemble pruning, semi-definite programming, heuristics, knowledge sharing

## 1. Introduction

Ensemble methods are gaining more and more attention in the machine-learning and data-mining communities. By definition, an ensemble is a group of learning models whose predictions are aggregated to give the final prediction. It is widely accepted that an ensemble is usually better than a single classifier given the same amount of training information. A number of effective ensemble generation algorithms have been invented during the past decade, such as bagging (Breiman, 1996), boosting (Freund and Schapire, 1996), arcing (Breiman, 1998) and random forest (Breiman, 2001). The effectiveness of the ensemble methods relies on creating a collection of diverse, yet accurate learning models.

Two costs associated with ensemble methods are that they require much more memory to store all the learning models, and it takes much more computation time to get a prediction for an unlabeled data point. Although these extra costs may seem to be negligible with a small research data set, they may become serious when the ensemble method is applied to a large scale real-world data set. In fact, a large scale implementation of ensemble learning can easily generate an ensemble with thousands of learning models (Street and Kim, 2001;

Chawla et al., 2004). For example, ensemble-based distributed data-mining techniques enable large companies (like WalMart) that store data at hundreds of different locations to build learning models locally and then combine all the models for future prediction and knowledge discovery. The storage and computation time will become non-trivial under such circumstances.

In addition, it is not always true that the larger the size of an ensemble, the better it is. For example, the boosting algorithm focuses on those training samples that are misclassified by the previous classifier in each round of training and finally squeezes the training error to zero. If there is a certain amount of noise in the training data, the boosting ensemble will overfit (Opitz and Maclin, 1999; Dietterich, 2000). In such cases, it will be better to reduce the complexity of the learning model in order to correct the overfitting, like pruning a decision tree. For a boosting ensemble, selecting a subset of classifiers may improve the generalization performance.

Ensemble methods have also been applied to mine streaming data (Street and Kim, 2001; Wang et al., 2003). The ensemble classifiers are trained from sequential chunks of the data stream. In a time-evolving environment, any change in the underlying data-generating pattern may make some of the old classifiers obsolete. It is better to have a screening process that only keeps classifiers that match the current form of the drifting concept. A similar situation occurs when classifiers are shared among slightly different problem domains. For example, in a peer-to-peer spam email filtering system, each email user can introduce spam filters from other users and construct an ensemble-filter. However, because of the difference of interest among email users, sharing filters indiscriminately is not a good solution. The sharing system should be able to pick filters that fit the individuality of each user.

All of the above reasons motivate the appearance of various ensemble pruning algorithms. A straightforward pruning method is to rank the classifiers according to their individual performance on a held-out test set and pick the best ones (Caruana et al., 2004). This simple approach may sometimes work well but is theoretically unsound. For example, an ensemble of three identical classifiers with 95% accuracy is worse than an ensemble of three classifiers with 67% accuracy and least pairwise correlated error (which is perfect!). Margineantu and Dietterich (1997) proposed four approaches to prune ensembles generated by Adaboost. KL-divergence pruning and Kappa pruning aim at maximizing the pairwise difference between the selected ensemble members. Kappa-error convex hull pruning is a diagram-based heuristic targeting a good accuracy-divergence trade-off among the selected subset. Back-fitting pruning is essentially enumerating all the possible subsets, which is computationally too costly for large ensembles. Prodromidis et al. invented several pruning algorithms for their distributed data mining system (Prodromidis and Chan, 2000; Chan et al., 1999). One of the two algorithms they implemented is based on a diversity measure they defined, and the other is based on class specialty metrics. The major problem with the above algorithms is that when it comes to optimizing some criteria of the selected subset, they all resort to greedy search, which is on the lower end of optimization techniques and usually without either theoretical or empirical quality guarantees. Kim et al. used an evolutionary algorithm for ensemble pruning and it turned out to be effective (Kim et al., 2002). A similar approach can also be found in (Zhou et al., 2001).

Unlike previous heuristic approaches, we formulate the ensemble pruning problem as a quadratic integer programming problem to look for a subset of classifiers that has the opti-

mal accuracy-diversity trade-off. Using a state-of-the-art semi-definite programming (SDP) solution technique, we are able to get a good approximate solution efficiently, although the original problem is NP-hard. In fact, SDP is not new to the machine learning and data mining community. It has been used for problems such as feature selection (d'Aspremont et al., 2004) and kernel optimization (Lanckriet et al., 2004). Our new SDP-based ensemble pruning method is tested on a number of UCI repository data sets with Adaboost as the ensemble generation technique and compares favorably to two other metric-based pruning algorithms: diversity-based pruning and Kappa pruning. The same subset selection procedure is also applied to a classifier sharing study. In that study, classifiers trained from different but closely related problem domains are pooled together and then a subset of them is selected and assigned to each problem domain. Computational results show that the selected subset performs as well as, and sometimes better than, including all elements of the ensemble.

Ensemble pruning can be viewed as a discrete version of weight-based ensemble optimization. The more general weight-based ensemble optimization aims to improve the generalization performance of the ensemble by tuning the weight on each ensemble member. If the prediction target is continuous, derivative methods can be applied to obtain the optimal weight on each ensemble model (Krogh and Vedelsby, 1995; Zhou et al., 2001; Hashem, 1997). In terms of classification problems, approximate mathematical programs are built to look for good weighting schemes (Demiriz et al., 2002; Wolpert, 1992; Mason et al., 1999). Those optimization approaches are effective in performance enhancement according to empirical results and are sometimes able to significantly reduce the size the ensemble when there are many zeros in the weights (Demiriz et al., 2002). However, size-reduction is not explicitly built into those programs and there is thus no control over the final size of the ensemble. The proposed ensemble pruning method distinguishes from the above methods by explicitly constraining the weights to be binary and using a cardinality constraint to set the size of the final ensemble. The goal of ensemble pruning is to contain the size of the ensemble without compromising its performance, which is subtly different from that of general weight-based ensemble optimization.

The rest of the paper is organized as follows. Section 2 describes the pruning algorithm in detail, including the mathematical formulation and the solution technique. Section 3 shows the experimental results on the UCI repository data sets and compares our method with other pruning algorithms. Section 4 is devoted to the algorithm's application in a classifier-sharing case study with a direct marketing data set. Section 5 concludes the paper.

## 2. Problem Formulation and Solution Technique

As the literature has shown, a good ensemble should be composed of classifiers that are not only accurate by themselves, but also independent of each other (Krogh and Vedelsby, 1995; Margineantu and Dietterich, 1997; Breiman, 2001), or in other words, they should make different errors. Some previous work has demonstrated that making errors in an uncorrelated manner leads to a low error rate (Hansen and Salamon, 1990; Perrone and Cooper, 1993). The individual accuracy and pairwise independence of classifiers in an ensemble are often referred to as strength and divergence of the ensemble. Breiman (2001)

showed that the generalization error of an ensemble is loosely bounded by $\frac{\bar{\rho}}{s^2}$ , where $\bar{\rho}$ is the average correlation between classifiers and $s$ is the overall strength of the classifiers. For continuous prediction problems, there are even closed-form representations for the ensemble generalization performance based on individual error and diversity. Krogh and Vedelsby (Krogh and Vedelsby, 1995) showed that for a neural network ensemble, the generalization error

$$E = \bar{E} - \bar{A},$$

where $\bar{E}$ is the weighted average of the error of the individual networks and $\bar{A}$ is the variance among the networks. Zhou et al. (2001) give another form,

$$E = \sum_{i,j} C_{ij},$$

where

$$C_{ij} = \int p(x)\Big(f_i(x) - d(x)\Big)\Big(f_j(x) - d(x)\Big)\,dx,$$

$p(x)$ is the density of input $x$, $f_i(x)$ is the output of the $i$th network and $d(x)$ is the true output. Note that $C_{ii}$ is the error of the $i$th network and $C_{ij,\,i\neq j}$ is a pairwise correlation-like measurement.

The problem is that the more accurate the classifiers are, the less different they become. Therefore, there must be a trade-off between the strength and the divergence of an ensemble. What we are looking for is a subset of classifiers with the best trade-off so that the generalization performance can be optimized.

In order to get the mathematical formulation of the ensemble pruning problem, we need to represent the error structure of the existing ensemble in a nice way. Unlike the case of continuous prediction, there is no exact closed-form representation for the ensemble error in terms of strength and diversity for a discrete classification problem. However, we are still able to obtain some approximate metrics following the same idea. From the error analysis of continuous problems, we notice that the ensemble error can be represented by a linear combination of the individual accuracy terms and pairwise diversity terms. Therefore, if we are able to find strength and diversity measurements for a classification ensemble, a linear combination of them should serve as a good approximation of the overall ensemble error. Minimizing this approximate ensemble error function will be the objective of the mathematical programming formulation.

First, we record the misclassifications of each classifier on the training set in the error matrix $P$ as follows:

$$\begin{aligned} P_{ij} &= 0, \quad \text{if } j\text{th classifier is correct on data point } i, \\ P_{ij} &= 1, \quad \text{otherwise.} \end{aligned} \tag{1}$$

Let $G = P^T P$. Thus, the diagonal term $G_{ii}$ is the total number of errors made by classifier $i$ and the off-diagonal term $G_{ij}$ is the number of common errors of classifier pair $i$ and $j$. To put all the elements of the $G$ matrix on the same scale, we normalize them by

$$\begin{aligned} \tilde{G}_{ii} &= \frac{G_{ii}}{N}, \\ \tilde{G}_{ij,\,i\neq j} &= \frac{1}{2}\left(\frac{G_{ij}}{G_{ii}} + \frac{G_{ij}}{G_{jj}}\right), \end{aligned} \tag{2}$$

where $N$ is the number of training points. After normalization, all elements of the $\tilde{G}$ matrix are between 0 and 1. $\tilde{G}_{ii}$ is the error rate of classifier $i$ and $\tilde{G}_{ij}$ measures the overlap of errors between classifier pair $i$ and $j$. Note that $\frac{G_{ij}}{G_{ii}}$ is the conditional probability that classifier $j$ misclassifies a point, given that classifier $i$ does. Taking the average of $\frac{G_{ij}}{G_{ii}}$ and $\frac{G_{ij}}{G_{jj}}$ as the off-diagonal elements of $\tilde{G}$ makes the matrix symmetric. The constructed $\tilde{G}$ matrix captures both the strength (diagonal elements) and the pairwise divergence (off-diagonal elements) of the ensemble classifiers. It is self-evident that for a good ensemble, all elements of the $\tilde{G}$ matrix should be small. Intuitively, $\sum_i \tilde{G}_{ii}$ measures the overall strength of the ensemble classifiers and $\sum_{ij,i\neq j} \tilde{G}_{ij}$ measures the diversity. A combination of these two terms, $\sum_{ij} \tilde{G}_{ij}$ should be a good approximation of the ensemble error. The diversity term defined here is ad hoc. We have noticed that there exist many other heuristic pairwise measurements for ensemble diversity. For instance, the disagreement measure (Ho, 1998), $\kappa$ statistic (Fleiss, 1981), Yule's $Q$ statistic (Yule, 1900), and so on. However, we found that the choice of the diversity measurement did not make a significant difference in terms of performance according to our computational experiments. Therefore, we stick to our definition because of its simplicity and intuitive appeal.

Now we can formulate the subset selection problem as a quadratic integer programming problem. Essentially, we are looking for a fixed-size subset of classifiers, with the sum of the corresponding elements in the $\tilde{G}$ matrix minimized. The mathematical programming formulation is as follows,

$$\begin{aligned} \min_{x} \quad & x^T \tilde{G} x \\ \text{s.t.} \quad & \sum_i x_i = k, \\ & x_i \in \{0,\ 1\}. \end{aligned} \tag{3}$$

The binary variable $x_i$ represents whether the $i$th classifier will be chosen. If $x_i = 1$, which means that the $i$th classifier is included in the selected set, its corresponding diagonal and off-diagonal elements will be counted in the objective function. Note that the cardinality constraint $\sum_i x_i = k$ is mathematically important because without it, there is only one trivial solution to the problem with none of the classifiers picked. In addition, it gives us control over the size of the selected subset.

This quadratic integer programming problem is a standard 0-1 optimization problem, which is NP-hard in general. Fortunately, we found that this formulation is close to that of the so-called "max cut with size $k$" problem (written MC-$k$), in which one partitions the vertices of an edge-weighted graph into two sets, one of which has size $k$, so that the total weight of edges crossing the partition is maximized. The MC-$k$ problem can be formulated as

$$\begin{aligned} \max_{y} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j) \\ \text{s.t.} \quad & \sum_i y_i = N_v - 2k, \\ & y_i \in \{-1,\ 1\}. \end{aligned} \tag{4}$$

where $N_v$ is the number of the vertices in the graph.

Roughly speaking, this optimization involves partitioning the vertices via the assignment of $y_i = 1$ or $y_i = -1$ to each vertex $i$ in the graph (subject to the size requirement) and minimizing the sum $\sum_{ij} w_{ij} y_i y_j$, where $w_{ij}$ is the edge weight between vertices $i$ and $j$. Notice that the interaction term $y_i y_j$ equals $-1$ when $i$ and $j$ are in different sets of the partition, which, in the context of the minimization described, contributes to the maximization of edges crossing the partition. The MC-$k$ problem is known to have a very good approximate solution algorithm based on semi-definite programming (SDP). The key point of the SDP approximation algorithm is to relax each binary variable $y_i \in \{-1, 1\}$ into a unit vector. Therefore, if we are able to transform the ensemble pruning formulation so that it fits into the framework of MC-$k$, we may obtain a good solution for the ensemble pruning problem.

The above MC-$k$ formulation (4) is equivalent to

$$\begin{aligned} \min_{y} \quad & y^T W y \\ \text{s.t.} \quad & \sum_i y_i = N_v - 2k, \\ & y_i \in \{-1, 1\}. \end{aligned} \tag{5}$$

where $W$ is the edge-weight matrix, with $w_{i,i} = 0$. If we compare this formulation with that of the ensemble pruning problem, the only barrier that prevents the application of the SDP approximation algorithm on the ensemble pruning problem is the difference in the possible values of the binary variable. Specifically, in ensemble pruning, $x_i \in \{0, 1\}$ and in MC-$k$, $y_i \in \{-1, 1\}$. Therefore, we need to make a transformation of variables for the ensemble pruning problem. Let

$$x_i = \frac{v_i + 1}{2}, \tag{6}$$

and $v_i \in \{-1, 1\}$. Now the objective function becomes

$$\frac{1}{4}(v + e)^T \tilde{G}(v + e), \tag{7}$$

where $e$ is a column vector of all 1s. The cardinality constraint $\sum_i x_i = k$ can be rewritten into quadratic form

$$x^T I x = k, \tag{8}$$

where $I$ is the identity matrix. After variable transformation, this constraint becomes

$$(v + e)^T I (v + e) = 4k. \tag{9}$$

A variable expansion trick can be applied to put both the transformed objective function and the cardinality constraint back into a nice quadratic form. We expand the variable vector $v = (v_1, v_2, ..., v_n)$ into $v = (v_0, v_1, v_2, ..., v_n)$, and let $v_0 = 1$. We then construct a new matrix

$$H_{(n+1)\times(n+1)} = \begin{pmatrix} e^T \tilde{G} e & e^T \tilde{G} \\ \tilde{G} e & \tilde{G} \end{pmatrix}. \tag{10}$$

Thus the objective function is equivalent to $v^T H v$ (dropping the coefficient). We use the same trick to construct a matrix D

$$D_{(n+1)\times(n+1)} = \begin{pmatrix} n & e^T \\ e & I \end{pmatrix}, \tag{11}$$

so that the cardinality constraint becomes $v^T D v = 4k$.

After this whole transformation, the problem formulation becomes

$$\begin{aligned} \min_{v} \quad & v^T H v \\ \text{s.t.} \quad & v^T D v = 4k, \\ & v_0 = 1, \\ & v_i \in \{-1,\ 1\},\ \forall i \neq 0, \end{aligned} \tag{12}$$

with which we will show later that the SDP relaxation applicable to the MC-$k$ formulation (5) (Goemans and Williamson, 1995; Han et al., 2002) may also be applied here.

It is important to keep in mind that the equivalence of our pruning problem and MC-$k$, established through the transformations (4–5) and (6–12), is an equivalence between optimal solution sets, not optimal values. Said differently, the optimal solutions of our problem are in one-to-one correspondence with the optimal solutions of MC-$k$ via the given transformations, but the optimal values of the two problems are not equal. In particular, even though we can interpret (12) as an instance of MC-$k$ with nonnegative weights on the complete graph, the objective function of (12) does not measure the total weight of cut edges. Instead, it more closely measures the total weight of un-cut edges (which is consistent with minimization) and, further, differs by a scaling as well as the inclusion of a constant term, which is based on the diagonal entries of $H$. These differences in the objective functions are byproducts of the transformations employed.

Given that our problem is equivalent to MC-$k$ (in terms of optimal solutions), it is worthwhile to ask what is known about MC-$k$. Based on the seminal work of Goemans and Williamson (1995) for the maximum cut problem with no size restriction, the papers of Feige and Langberg (2001) and Han, Ye, and Zhang (2002) demonstrate an approximation algorithm for instances of MC-$k$ with nonnegative edge weights. The approximation algorithm is based on solving a SDP relaxation of MC-$k$ and applying a simple-to-implement randomization scheme.

Because the objective function of our problem does not precisely match the objective of MC-$k$ (even though the two problems are equivalent through optimal solutions), the approximation guarantee for MC-$k$—which is with respect to its objective function—does not necessarily transform into a guarantee for our problem. In fact, it seems extremely difficult to derive directly any approximation guarantees for our problem. SDP-based approximation approaches have proven most successful for problems in maximization form due to technical details concerning how an SDP relaxes a binary quadratic program, and unfortunately, our problem is in minimization form.

Nevertheless, we feel that the strong connection of our problem with MC-$k$ justifies the following heuristic procedure: (i) transform an instance of our problem into an instance of MC-$k$; (ii) use the SDP-based approximation algorithm to obtain a good solution of the

MC-$k$ instance; and (iii) transform back to a solution of our problem. Further, it is not difficult to see that the above three steps are equivalent to the more direct approach of relaxing (12) as an SDP and applying the randomization scheme of Feige and Langberg (2001) and Han, Ye, and Zhang (2002). In other words, it is not explicitly necessarily to convert to an instance of MC-$k$ first.

For completeness, we now return our attention to the SDP relaxation itself. Problem (12) is equivalent to

$$\begin{array}{ll} \min\limits_{v} & H \bullet vv^T \\ \text{s.t.} & D \bullet vv^T = 4k, \\ & v_0 = 1, \\ & v_i \in \{-1,\ 1\},\ \forall i \neq 0, \end{array} \tag{13}$$

where $A \bullet B = \sum_{i,j} A_{ij}B_{ij}$.

To construct the relaxation, we first note that the constraint $v_0 = 1$ can be relaxed to $v_0 \in \{-1, 1\}$ without changing the problem since $-v$ is feasible for the remaining constraints if and only if $v$ is and since $H \bullet vv^T = H \bullet (-v)(-v)^T$. Next, we rewrite the constraints $v_i \in \{-1, 1\}$, $i = 0, 1, \ldots, n$ as the single, collective constraint $\text{diag}(vv^T) = e$ to arrive at the following formulation:

$$\begin{array}{ll} \min\limits_{v} & H \bullet vv^T \\ \text{s.t.} & D \bullet vv^T = 4k, \\ & \text{diag}(vv^T) = e. \end{array} \tag{14}$$

We next substitute $V = vv^T$, and note that $V$ can be expressed as $vv^T$ if and only if $V \succeq 0$ with $\text{rank}(V) = 1$, which gives us

$$\begin{array}{ll} \min\limits_{V} & H \bullet V \\ \text{s.t.} & D \bullet V = 4k, \\ & \text{diag}(V) = e \\ & V \succeq 0, \quad \text{rank}(V) = 1. \end{array} \tag{15}$$

Although this problem is written in a different form, it is completely equivalent to our original 0-1 quadratic problem.

The SDP relaxation is now obtained by dropping the rank constraint, which yields the following (convex) SDP:

$$\begin{array}{ll} \min\limits_{V} & H \bullet V \\ \text{s.t.} & D \bullet V = 4k, \\ & \text{diag}(V) = e \\ & V \succeq 0. \end{array} \tag{16}$$

Now the original NP-hard problem (3) is relaxed into a convex SDP problem which can be solved to any preset precision in polynomial time. We solve the SDP relaxation using the publicly available package SDPLR (Burer and Monteiro, 2003; SDPLR) and have implemented the approximation algorithm described in (Han et al., 2002).

## 3. Computational Experiments

The SDP-based ensemble pruning algorithm is tested on sixteen UCI repository data sets (Blake and Merz, 1998): Autompg, Bupa, Cmc, Crx, Glass, Haberman, Housing, Cleveland-heart-disease, Hepatitis, Ion, Pima, Sonar, Vehicle, WDBC, Wine and WPBC. Some of the data sets do not originally depict two-class problems so we did some transformation on the dependent variables to get binary class labels. Specifically in our experiments, the Autompg data is labeled by whether the mileage is greater than 25mpg, the Housing data by whether the value of the house exceeds $\$25,000$, and the Cleveland-heart-disease by the presence or absence of the disease. Vehicle and Wine are multi-class problems so we set the problem as separating one class pair each time, resulting in a total of 24 data sets. All data points with missing values are removed.

It has been shown that the pruning effect is more striking on a diverse ensemble (Margineantu and Dietterich, 1997). Boosting-like algorithms usually generate diverse classifiers by concentrating on widely different parts of the training samples at each round of training. So we use Adaboost to create the original ensemble for pruning. The unpruned C4.5 decision tree is used as the base classifier training algorithm (Quinlan, 1993). One hundred decision trees are built for each data set following the standard Adaboost procedure. If the training error reaches zero before the number of trees gets 100, the Adaboost process is repeated (with different random seeds). Note that the accuracy of all the classifiers created by Adaboost is higher than 50%.

Two existing metric-based ensemble pruning algorithms, diversity-based pruning and kappa pruning, are picked as the benchmarks because the objectives of the these two algorithms are somewhat close to that of the new algorithm. In addition, these two algorithms can prune the ensemble to any pre-set size, which makes a fair comparison possible. Diversity-based pruning tries to maximize the sum of pairwise diversity of the selected subset of classifiers (Prodromidis and Stolfo, 1998). Pairwise diversity is defined as the proportion of the training points on which the two classifiers disagree. The greedy search starts with the most accurate classifier and adds the classifier that improves the objective most at each round. Kappa-pruning, invented by Margineantu and Dietterich (Margineantu and Dietterich, 1997), attempts to minimize the sum of pairwise similarity of the selected subset. A $\kappa$ statistic is used to measure the pairwise similarity,

$$\kappa = \frac{\theta_1 - \theta_2}{1 - \theta_2}$$

where $\theta_1$ is the proportion of points on which the two classifiers agree with each other on the training set, and $\theta_2$ is the probability that the two classifiers agree purely by chance. The $\kappa$ statistics among all pairs are calculated and ranked from low to high. The greedy search picks classifier pairs from the ranked pair list until the pre-set size of the pruned ensemble is met.

The performance of the three algorithms on the 24 data sets are listed in Table 3. Here, the size of the pruned ensemble is 25. Empirical research suggests that, in most cases, most or all of the generalization gain in a well-constructed ensemble comes from the first 25 classifiers added (Breiman, 1996; Opitz and Maclin, 1999). The result is averaged over five ten-fold cross-validations. The number in the parentheses is the standard deviation over the five runs. A win-loss-tie summarization based on mean value and $t$ test (95%

| dataset | SDP-25 | Div-25 | Kappa-25 | No Pruning |
|---|---|---|---|---|
| AUTOMPG | 10.35(0.62) | 13.88(0.83) | 11.91(1.52) | 10.71(0.70) |
| BUPA | 30.52(1.43) | 35.87(1.25) | 38.39(2.65) | 30.32(0.43) |
| CMC | 32.82(0.87) | 41.70(1.66) | 43.66(1.37) | 34.50(1.19) |
| CRX | 13.88(0.46) | 22.40(3.41) | 21.78(4.44) | 13.58(0.85) |
| GLASS | 12.53(1.12) | 17.30(2.88) | 16.39(2.76) | 11.29(0.70) |
| HABERMA | 32.73(2.21) | 38.63(1.30) | 38.88(3.09) | 34.56(1.74) |
| HEART | 20.13(2.19) | 28.09(2.73) | 27.81(4.09) | 20.40(2.39) |
| HEPATIT | 15.81(1.17) | 19.83(2.66) | 16.86(2.00) | 14.71(1.84) |
| HOUSING | 11.03(0.61) | 12.37(0.91) | 12.21(1.59) | 10.67(0.66) |
| ION | 7.46(2.17) | 10.94(5.13) | 14.02(10.61) | 9.85(7.72) |
| IRIS | 12.20(9.47) | 15.00(7.84) | 19.60(8.44) | 13.40(11.01) |
| PIMA | 25.34(0.67) | 31.31(3.78) | 30.24(2.90) | 25.06(0.78) |
| SONAR | 21.43(1.68) | 26.24(4.32) | 25.24(2.31) | 18.96(1.13) |
| WDBC | 3.38(0.34) | 3.51(0.59) | 3.76(0.76) | 2.88(0.30) |
| WINE1-2 | 2.92(0.64) | 6.15(3.17) | 12.62(18.45) | 11.85(18.82) |
| WINE1-3 | 1.11(0.76) | 1.27(0.50) | 2.58(1.01) | 1.31(0.47) |
| WINE2-3 | 3.05(0.75) | 4.56(2.50) | 5.67(6.27) | 2.86(3.54) |
| WPBC | 24.06(2.51) | 32.35(1.88) | 29.54(3.52) | 24.04(2.79) |
| VEHICLE1-2 | 41.15(2.62) | 42.32(1.01) | 45.17(4.65) | 41.40(1.28) |
| VEHICLE1-3 | 1.07(0.42) | 3.67(3.79) | 9.26(12.11) | 3.21(3.24) |
| VEHICLE1-4 | 4.52(0.36) | 6.47(1.24) | 6.18(2.42) | 3.84(0.21) |
| VEHICLE2-3 | 2.25(0.55) | 7.34(4.01) | 11.99(7.06) | 5.82(4.36) |
| VEHICLE2-4 | 5.00(1.28) | 10.33(3.26) | 13.57(12.65) | 5.96(4.35) |
| VEHICLE3-4 | 1.15(0.11) | 0.96(0.41) | 1.39(1.63) | 0.67(0.39) |
| | Absolute W-L-T | 23-1-0 | 24-0-0 | 12-12-0 |
| | Significant W-L-T | 9-0-15 | 8-0-16 | 2-0-22 |

Table 1: Comparison of SDP pruning, Diversity-based pruning, Kappa-pruning and original ensembles, by % error and (standard deviation)

significance level) is attached at the bottom of the table. Note that simple majority voting is used to combine the predictions of the ensembles. Prodromidis et al. (Prodromidis and Stolfo, 1998; Prodromidis and Chan, 2000) built a higher level classifier (meta-classifier) to aggregate the output of the pruned ensembles. There has been other research in this direction (Wolpert, 1992; Bennett et al., 2000; Mason et al., 1999; Grove and Schuurmans, 1998). However, there is so far no strong evidence that such a meta-classifier is generally better than simple majority voting.

Table 3 shows that the performance of the SDP-based pruning is better than that of the other two algorithms for most of the data sets involved in the computational experiments. Also, although only a quarter of the classifiers are left, the error of the pruned ensemble by SDP-based pruning is statistically the same as that of the original ensemble. In addition, we may conclude that the SDP-based pruning is more stable in terms of accuracy, by looking at the error standard deviation of the three algorithms. The error fluctuation range of the other two algorithms is often much larger, which might explain why the SDP-based pruning is sometimes not statistically better.

There are several possible reasons why the SDP-based pruning outperforms the other two. First, the optimization procedure of the SDP-based pruning is better than greedy search. Although greedy search may produce optimal solutions in some cases, it may also perform badly when the problem is ill-conditioned, which is not a rare case for ensemble pruning. On the other hand, while the SDP approximation algorithm may not be able find optimal solutions all the time, it can provide a relatively good solution in most cases. Therefore, the SDP-based pruning turns out to be better on average and more stable. Second, the definition of divergence of the SDP-based pruning is subtly different from that of the two others. SDP-based pruning only considers the correlation between errors while the other two define pair-wise correlation based on all the data points. In fact, we want the classifiers to agree with each other when they make correct predictions. The divergence we need is indeed the divergence of the way they commit errors. Hence, the error-based divergence definition is more closely related to generalization performance. Finally, the objective function of the SDP-based pruning may reflect a better trade-off between individual accuracy and pair-wise divergence. As a matter of fact, the other two algorithms do not explicitly include individual accuracy in their objective functions. The diversity-based pruning starts the search with the most accurate classifier, and both favor more accurate classifiers when the diversity measures tie. For a systematically constructed ensemble, the individual accuracy of each classifier is not critical as long as its accuracy is over 50% and there are enough of classifiers in the ensemble. However, for a pruned ensemble, with the initial structure broken and a large portion of the members removed, the importance of individual accuracy may increase. Therefore, the individual strength of the classifiers should be appropriately considered in ensemble pruning. Although our way of including individual accuracy in the objective may not be perfect, it may contribute to the improvement of performance.

The improved performance comes with a price: more computation time. Table 3 shows the time it takes to solve the pruning problem with increasing size of the original ensemble. The size of the pruned ensemble is fixed at 25. The timing test was conducted on an AMD 1.2GHz computer with 1G memory. As we mentioned before, with the SDP relaxation heuristic, the pruning problem can be solved in polynomial time. This can be verified by looking at the linearity of the $\log \log$ plot of time vs. size of the original ensemble, as illustrated in Figure 1. Selecting 25 classifiers from an original ensemble with 3000 classifiers will take roughly a day, which is within the reasonable region for most applications. If each classifier (a decision tree, e.g.) is trained with 1000 points, such an ensemble learning procedure would be able to handle a dataset with 3,000,000 points. Therefore, we may cautiously conclude that the SDP-based ensemble pruning method is applicable to large scale data mining problems. Note that our subset selection routine is coded partly in MATLAB. If it were coded in C, it would have been much faster.

## 4. Selective Sharing of Classifiers Among Related Problem Domains: A Case Study

We have so far demonstrated the effectiveness of the SDP-based pruning algorithm based on small research data sets. As mentioned before, ensemble pruning is more relevant to large-scale data mining implementations. Therefore, we wish to demonstrate that the SDP-based pruning algorithm is also effective and efficient under real-world conditions. We imple-

| Size | Time (s) | Size | Time (s) |
|---|---|---|---|
| 100 | 3 | 1500 | 4889 |
| 200 | 36 | 2000 | 23255 |
| 400 | 113 | 2500 | 42419 |
| 800 | 713 | 3000 | 83825 |

Table 2: Computation time of SDP-based pruning

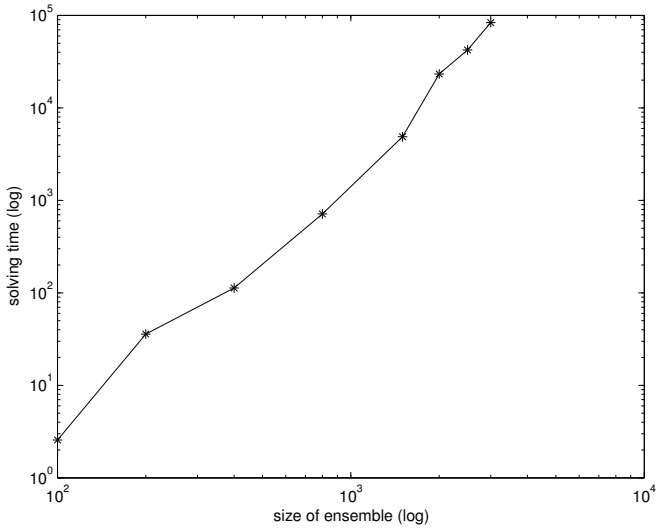


Figure 1: Timing test of SDP-based ensemble pruning

mented the pruning algorithm in the following case study, where classifiers from different but closely-related problem domains are pooled together and a subset of them is then selected for each problem domain. This case study involves a much larger data set than the UCI sets used in the above experiments, and an original ensemble with a larger number of more divergent classifiers. The results of the study verify the algorithm's performance in real-world applications, and show one type of situation where ensemble pruning can be particularly useful.

As we know, ensemble methods not only improve classification accuracy, but also provide a way to share knowledge. If the data for a problem domain are distributed at different locations, classifiers can be trained locally and then combined to create an ensemble. This centralized ensemble gathers information from each site and can potentially be more powerful for further predictions. Now the questions is, if we are working with several different but closely related problem domains, is sharing classifiers among those domains still a good idea?

The essence of sharing classifiers is sharing common knowledge among different but closely related problem domains. A famous example of the research in this direction is the multi-task neural network (Caruana, 1997). Each problem domain is assigned one or more output nodes, while sharing some of the input and mid-layer nodes with other problem domains. Although this method is sometimes successful, it has several limitations. First, it requires that the data be at the central location for training. Second, the training speed becomes a big issue if the size of the data is large. The classifier-sharing strategy may avoid both of the drawbacks. Since the classifiers can be trained locally, the data need not be centralized. Moreover, one can use efficient algorithms like decision trees to train classifiers, so computation time becomes less of a problem.

The prerequisite for sharing classifiers among different problem domains is that the data schema for each problem domain should be identical, or at least very similar. It ensures that the classifiers trained on one problem domain can also be applied to other problem domains, although the accuracy can possibly be low. Sometimes, this requires transformation of variables to satisfy this condition. It is in fact hard to tell a priori whether sharing classifiers among different problem domains will improve the overall performance. The success of this strategy depends on the connections among the problem domains and the self-completeness of information within each problem domain. However, with a screening process (ensemble pruning) that will be described later, the chance that sharing classifiers will eventually downgrade the overall performance is minimal.

The cross-domain classifier-sharing strategy is tested on a publicly available marketing data set. To address the concern that sharing classifiers blindly sometimes may do harm to some of the problem domains if there exist conflicting elements among them, we apply the SDP-based pruning algorithm to select a good subset of classifiers from the entire ensemble for each problem domain.

The data set is a catalog marketing data set from the Direct Marketing Association (DMEF Academic Data Set Three, Specialty Catalog Company, Code 03DMEF). The dependent variables are the customers' responses to a particular promotion held by a catalog company. There are 19 different categories of products involved in the promotion, which correspond to 19 response variables. Unfortunately, the identities of the product categories are not available. The data mining problem we try to solve is to predict which categories

| Category | %Pos | Category | %Pos |
|---|---|---|---|
| 1 | 0.14 | 11 | 0.13 |
| 2 | 0.27 | 12 | 0.05 |
| 3 | 0.04 | 13 | 0.44 |
| 4 | 0.92 | 14 | 2.65 |
| 5 | 0.12 | 15 | 1.70 |
| 6 | 0.83 | 16 | 2.09 |
| 7 | 0.44 | 17 | 0.65 |
| 8 | 0.37 | 18 | 0.31 |
| 9 | 0.64 | 19 | 1.17 |
| 10 | 0.02 | | |

Table 3: Percentage of positive points (response rate) of 19 categories.

of products a customer is going to buy based on the available historical and demographic data. We decomposed this task into 19 separate binary classification problems, building one model for each category and predicting whether the customer is going to buy from that category. Note that this whole task cannot be treated as a classification problem with 19 classes because one customer can buy products from any number of categories, including zero, so the class assignment for each individual is not exclusive. This kind of problem is sometimes referred to as "multi-label" classification problem and is more often seen in the text-mining domain (McCallum, 1999; Shen et al., 2004).

Table 4 shows the percentage of positive points, or response rate, of each category. A positive point for a category represents a customer that buys one or more products from that category. It can be seen from the table that for most categories, the response rate is lower than one percent, which makes the data set highly skewed toward the non-buyers. A common way to deal with unbalanced data is to create training data sets with a balanced class distribution through sampling (Fan et al., 2004).

For each category, twenty five training sets with 400 positive points and 400 negatives are bootstrapped from the original training data. A C4.5 decision tree is then built based on each training set. In total, twenty-five decision trees are obtained for each category. These twenty-five trees are grouped together to create an ensemble for future predictions. This bagging-like approach is a standard way to solve such marketing problems and often very powerful. However, it does not work well on the marketing data set used in this study. For example, the lift curve for Category 10 is almost a 45 degree line, which implies that the ensemble learned hardly anything useful from the training sets. Our explanation is that the number of distinctive positive points in the training sets for Category 10 is too small. As shown in Table 4, the original response rate for category 10 is only 0.02%, so there is simply not enough information to build a decent classifier.

To improve the performance of the ensembles built by the original bagging approach, extra useful information is necessary. Sharing classifiers among different categories is our proposed solution. We cite two reasons to support this proposition. First, we are dealing with customers from the same catalog company. It is reasonable to expect that those customers who did place orders should share some common properties and those properties should be reflected in the classifiers belonging to those categories with relatively higher

response rate. If these classifiers can be included into the ensembles of those categories without enough positive points, they may help hit the targets and improve the overall performance. Second, the purchase patterns of some different categories may be similar. For example, people are more likely to buy clothes when there are discount coupons. This may also be true for shoes. Therefore, a marketing model for clothes that stresses the importance of discount coupons may also work for shoes although they belong to different product categories.

A naive way of sharing classifiers is to pool all the classifiers from the 19 categories into one big ensemble and use it for every category. However, there exist risks behind this sharing-all strategy. Specifically, when there are strikingly conflicting concepts among the problem domains, mixing classifiers blindly will degrade the effectiveness of the ensemble method. Therefore, it is safer to bring in the SDP-based screening process that is able to select a subset of classifiers for each problem domain. Unlike the experiments on the UCI data sets, here we use a held-out tuning set for each subset selection process since there is a large amount of data. Therefore, the $P$ matrix in (1) is constructed based on classification results on the tuning set instead of the training set. Each tuning set reflects the original class distribution. Note that there are 19 categories in the marketing data, so there will be 19 tuning sets and the subset selection process will be repeated 19 times, once for each category.

There is still one more problem with the current setup of the $\tilde{G}$ matrix. Since the tuning data sets here reflect the original class distributions, which are highly biased towards non-buyers, the resulting $\tilde{G}$ matrix will reflect the performance of the ensemble on the negative points while almost ignoring the influence of the positive points. It is necessary to balance the influence of the positive points and the negative points on $\tilde{G}$. To achieve this goal, we define a third $\hat{G}$ matrix as a convex combination of the $\tilde{G}$ matrix on the positive points and the $\tilde{G}$ matrix on the negative points in the tuning set,

$$\hat{G} = \lambda \tilde{G}_{pos} + (1 - \lambda)\tilde{G}_{neg}.$$

Note that $\tilde{G}_{pos}$ is computed based only on the positive points in the tuning set and $\tilde{G}_{neg}$ only on the negative points, using formulas (1) and (2). In the following computational experiments, $\lambda$ is set to 0.5.

The original data schema for each category is identical: same independent variables (purchase history, promotions and demographic information) and a binary dependent variable indicating whether the customer buys products from this category. However, we found that some of the independent variables are category-specific. For example, there are 19 variables each representing whether there is a discount for each category. Intuitively, the discount variable for Category $i$ is more informative to the prediction problem for Category $i$ than for other categories. There is likely a split on the discount variable for Category $i$ in the decision trees for Category $i$. Since this discount variable is probably (not absolutely) not relevant to other categories, the decision trees induced on Category $i$ are less likely to be useful for other categories. To make the decision trees more interchangeable among different categories, we did some transformations on those category-specific variables. In the data set for Category $i$, a copy of each category-specific variable related to Category $i$ is appended to the end of the data schema and labeled as "xxxx-for-this-category". The values of the original category-specific variables for this Category $i$ (which already have

| Cat. | Non-sharing | Naive-Sharing | Selective-sharing |
|---|---|---|---|
| 1 | 77.69(4.08) | 82.93(1.21) | 82.31(2.22) |
| 2 | 76.89(2.03) | 80.45(0.47) | 77.71(5.81) |
| 3 | 63.95(1.69) | 84.79(1.15) | 83.02(5.52) |
| 4 | 82.38(1.13) | 80.93(0.40) | 77.94(3.15) |
| 5 | 73.31(4.35) | 82.06(0.95) | 80.38(1.57) |
| 6 | 80.45(0.60) | 80.32(0.42) | 78.20(2.58) |
| 7 | 79.79(1.41) | 81.20(0.65) | 79.72(4.73) |
| 8 | 75.96(0.43) | 78.67(0.46) | 76.71(1.51) |
| 9 | 81.72(0.78) | 81.33(0.58) | 79.04(2.55) |
| 10 | 52.67(9.71) | 74.82(3.55) | 72.33(6.29) |
| 11 | 68.62(6.54) | 80.46(1.58) | 76.74(2.55) |
| 12 | 57.11(5.56) | 78.76(2.81) | 75.42(3.56) |
| 13 | 79.45(0.97) | 78.56(0.52) | 77.03(2.30) |
| 14 | 83.50(0.26) | 81.41(0.30) | 81.25(2.74) |
| 15 | 97.34(0.16) | 93.05(1.58) | 97.06(0.35) |
| 16 | 97.63(0.07) | 93.51(1.33) | 97.66(0.13) |
| 17 | 82.86(0.85) | 84.82(0.61) | 83.80(0.40) |
| 18 | 85.60(1.05) | 87.37(0.73) | 88.01(0.35) |
| 19 | 91.27(0.23) | 89.69(0.82) | 91.16(0.55) |
| | | **Comparison** | **Selective vs. other** |
| | 11-8-0 | 4-15-0 | Abs.W-L-T |
| | 3-0-16 | 3-1-15 | Sig.W-L-T |

Table 4: AUCs of non-sharing, naive-sharing and selective-sharing ensembles

copies) are then set to be uniform for each record so that there will be no splits on these variables in the decision trees. The splits, if necessary, will be made on those newly appended variables. After transformation, each category has the same number of appended category-specific variables related only to itself so the data schema of each category is still identical and the tree splits on the category-specific variables are more meaningful across categories. For instance, if there is a rule induced on a category like "if there is a discount on this category, the customer is going to buy products from this category", it is reasonably applicable to other categories. Therefore, the interchangeability of trees among categories is increased.

Since the original bagging approach builds ensembles of 25 classifiers, here we set the size of the selected ensemble to be 25 as well for a fair comparison. Another reason to set the size of the selected ensemble equal to that of the original ensemble is that for each category, there are at least 25 relevant classifiers: those that are trained on its own data. In fact, we have experimented with other sizes such as 50, but it didn't result in any significant difference for this particular problem.

Table 4 lists the AUCs (Area Under the ROC Curve) of three methods: original bagging (non-sharing), naive-sharing and selective-sharing, on each category. Summaries based on mean value and paired $t$ tests (95% significance level) are attached at the bottom of the table.

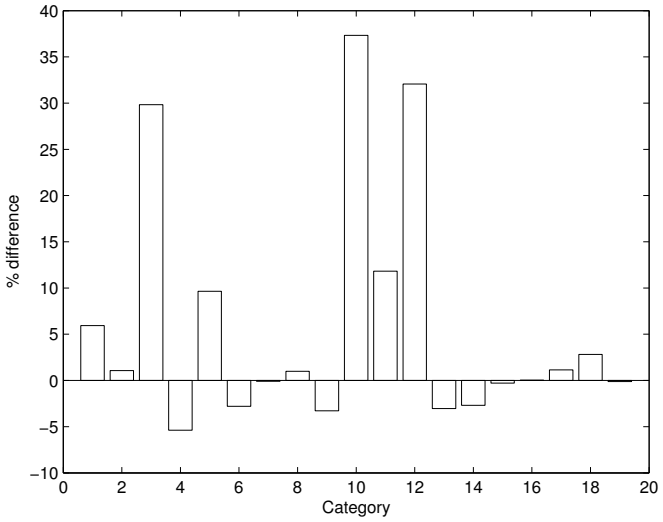


Figure 2: % Difference of AUCs of selective-sharing ensemble and the original bagging ensembles on 19 categories, $\frac{AUC_{sel}-AUC_{orig}}{AUC_{orig}}$

The results show that selective sharing of classifiers does improve the AUCs for almost half of the categories. Especially for those categories without enough positive information, the rise in AUC is substantial (Figure 2). The overall performance of the ensembles produced by selective-sharing is almost as good as the naive-sharing ensemble. For Categories 15, 16 and 19, it is even statistically better. Note that the selected ensembles use only 25 classifiers each as compared to 475 for the naive-sharing ensemble. There are two implications of this result. First, the ensemble pruning process is quite effective. Second, the reason that the including-all ensemble is better than the individual bagging ensembles is not simply because it's larger. There is truly useful additional information in the including-all ensemble that can be singled out by the pruning process.

Moreover, the selective sharing is a conservative version of sharing classifiers, hence it should be more robust. If there were wildly conflicting concepts among the categories, the selective-sharing would have performed better. In fact, the selective-sharing ensembles of Categories 15 and 16 do outperform the including-all ensemble by throwing away bad classifiers, as expected.

It is also interesting to look at the classifier sharing structure among different categories. Table 4 provides a list of statistics that summarizes the sharing structure, averaged over the same five runs in Table 4. The "Prior" column shows the response rate for each category. The "Used" column is the total number of classifiers trained on this category used for all categories. The "Used Own" column is the total number of classifiers trained on this category used for this category. The "Most Used" column shows the most common training category for the classifiers for each ensemble. Finally, HI index is the Herfindahl index

| **Category** | **Prior** (%) | **Used** | **Used Own** | **Most Used** | **HI index** |
|---|---|---|---|---|---|
| 1 | 0.14 | 2 | 1 | 13 | 0.16 |
| 2 | 0.27 | 0 | 0 | 14 | 0.27 |
| 3 | 0.04 | 0 | 0 | 6 | 0.16 |
| 4 | 0.92 | 12 | 1 | 17 | 0.25 |
| 5 | 0.12 | 1 | 0 | 17 | 0.27 |
| 6 | 0.83 | 17 | 0 | 14 | 0.35 |
| 7 | 0.44 | 9 | 0 | 14 | 0.35 |
| 8 | 0.37 | 14 | 1 | 14 | 0.67 |
| 9 | 0.64 | 22 | 3 | 14 | 0.29 |
| 10 | 0.02 | 1 | 0 | 19 | 0.33 |
| 11 | 0.13 | 3 | 0 | 19 | 0.57 |
| 12 | 0.05 | 2 | 0 | 19 | 0.47 |
| 13 | 0.44 | 18 | 0 | 19 | 0.67 |
| 14 | 2.65 | 80 | 7 | 14 | 0.30 |
| 15 | 1.70 | 11 | 0 | 19 | 0.33 |
| 16 | 2.09 | 14 | 1 | 19 | 0.35 |
| 17 | 0.65 | 94 | 9 | 19 | 0.34 |
| 18 | 0.31 | 24 | 2 | 19 | 0.49 |
| 19 | 1.17 | 150 | 18 | 19 | 0.63 |

Table 5: Statistics of classifier sharing structure

(Rose and Engel, 2002), which is computed by the following formula:

$$HI_i = \sum_{j=1}^{C} (\frac{n_{ij}}{N_e})^2$$

where $n_{ij}$ is the number of classifiers trained on $j$th category used for $i$th category, and $N_e$ is the size of the pruned ensemble which is 25 in this case. The smaller the HI index, the more diverse the ensemble, in terms of original training categories. From our observation, the classifier sharing structure is reasonably stable for most of the categories over the five runs. For instance, classifiers from Categories 14, 17 and 19 are always the top three used classifiers in the selected ensembles, the most used column seldom varies among different runs, etc.

There are a few things that one can tell after studying these numbers.

- The most surprising fact shown by the table is that for most categories, the classifiers trained on the category are seldom chosen for its own ensemble. Only Categories 14, 17 and 19 used a reasonable number of their own classifiers. However, the performance of the pruned ensembles are close to that of the original bagging ensembles as shown in Table 4. This may imply that some of the categories are closely related therefore their classifiers are highly interchangeable through combination.

- The higher the original response rate for each category, the more times its classifiers are used by other categories. Figure 3 plots the number of times its classifiers are used

versus the original response rate for each category. The slope of the linear regression is 24.00 with $p$ value 0.053, which verifies the conjecture that the classifiers trained on categories with more positive information may well capture the general properties of the potential customers and are thus widely used by other categories. The fact that the $p$ value is slightly below the 95% significance level implies that the response rate is not the sole factor that decides the popularity of the classifiers.

- Categories 15, 16 and 17 are the major outliers from the above rule. Categories 15 and 16 have relatively high response rate, but their classifiers are seldom used, even hardly used in their own ensembles. The including-all ensemble performs badly on these two categories. On the other hand, the pruned ensembles perform almost as well as the original bagging ensembles. Our explanation is that these two categories are a little different from the main trend. However, these two categories are somehow closely related to a subset of categories so that a combination of classifiers from those categories may still predict these two categories well. It is unclear why their own classifiers are so rarely used for their own categories. One guess is that the original bagging ensemble members for these two categories are not good individually, though they perform well as a group. So the pruning algorithm throws them away because they are not strong enough by themselves. Category 17 is an outlier from the other side. It has relatively low response rate, but its classifiers are the second most used. This indicates that despite the low response rate, the classifiers trained on Category 17 are still good or often provide information that is complementary to that provided from other categories. In addition, there is a subset of categories in which category 17 may be a good representative. This subset includes Categories 4 and 5 because these two categories used many classifiers from Category 17, as shown in the "Most Used" column of Table 4.

- Classifiers from categories with response rate less than 0.3% are hardly used. Usually, the including-all ensemble performs pretty well on those categories. It shows that for those problem domains without enough information, using aggregated information from all related problem domains may be a good solution.

- Classifiers from Categories 14 and 19 occupy almost half of the classifiers that are selected. Classifiers from Category 14 dominate pruned ensembles in six categories while classifiers from Category 19 dominate in nine categories. This might be because the customers of these two categories well-represent the potential buyers of the whole catalog. There exists a subtle difference between these two categories. For Category 14, although its classifiers are widely used, there are only 7 selected for its own problem, while for Category 19, almost all of its own classifiers are selected. There are two explanations of this phenomenon. First, classifiers of Category 14 captures well only part of the characteristics of its customers, therefore, it still needs extra information for its own problem. Second, those classifiers are good but very close to each other. So the divergence criteria of the subset selection process prevents including too many of its own classifiers.

- The Herfindahl index shows the homogeneity of the sources of the classifiers in the pruned ensembles. If a category prefers universal knowledge for its prediction, the

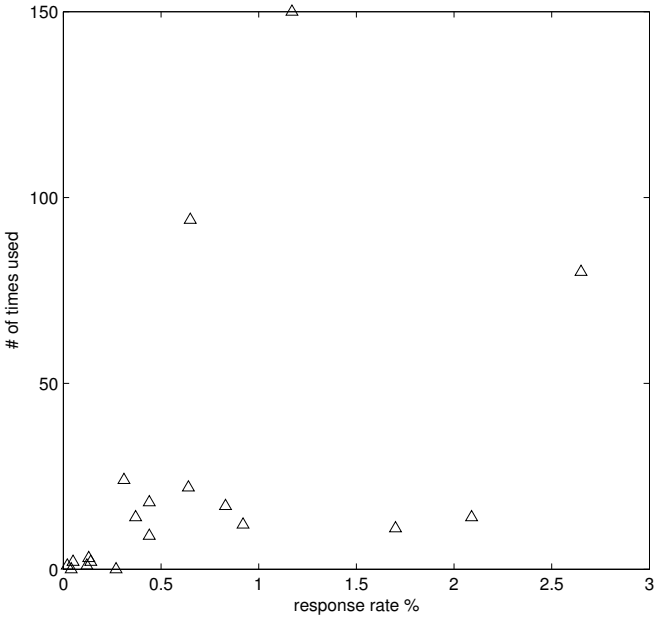


Figure 3: Response rate of each category vs. total number of times its classifiers are used in the selected ensembles

HI index will be low, which means that the pruned ensemble picks classifiers from many categories. On the other hand, if a category needs specialized knowledge for its prediction, the HI index will be higher, which implies that the pruned ensemble picks classifiers only from a small number of categories that share some particular properties with the category in question. Usually for those categories with small response rate, the HI index is low, for example Category 1 and Category 3. Categories 8 and 13 have the highest HI index. From the "Most Used" column, we know that Category 8 used classifiers mostly from Category 14. So it can be inferred that Category 8 and category 14 are closely related. For the same reason, the customers of Category 13 may share some special properties with that of Category 19.

These pieces of information might make more sense if we knew what these categories were. Unfortunately, this knowledge is currently unavailable. The information gained by studying the classifier sharing structure may help improve the mailing and promotion strategy of the catalog company. For instance, customers that buy products from Categories 14, 17 and 19 seem to capture the main trend of the customer base, so they are likely the core customers of the company and may need extra attention. Customers of Category 14 and 8 seem to be similar from some perspective. Therefore, a promotion strategy that proves to be successful for Category 14 may also work well for Category 8.

## 5. Conclusion and Future Work

This paper introduced a new ensemble pruning method to improve efficiency and effectiveness of an existing ensemble. Unlike previous heuristic approaches, we formulate the ensemble pruning problem as a strict mathematical programming problem and apply SDP relaxation techniques to obtain a good approximate solution. The computational experiments on the UCI repository data sets show that this SDP-based pruning algorithm performs better than two other metric-based pruning algorithms. Its application in a classifier sharing study also indicates that this subset selection procedure is effective in picking classifiers that fit the needs of different problem domains. Besides the peer-to-peer email filtering problem mentioned before, this method can also be useful when a company is trying to promote a new product. Usually, there is only limited information about a new product's potential customers. However, if the company has good marketing models for its old products, especially those closely related to the new product, selecting some of the old models based on the limited data of the new product may be a better solution than building models directly.

There is yet room for improvement in the current version of the algorithm. For example, there are several parameters in the model that can be fine-tuned, such as the method of normalization and the relative weight between the diagonal terms and off-diagonal terms. Another thing that's worth exploring is whether there exits a nice form of the objective function so that a "real" optimal subset can be found without enforcing the cardinality constraint. Under the current setup, removing the cardinality constraint will result in a trivial solution.

## Acknowledgment

Samuel Burer would like to acknowledge support from NSF Grant CCR-0203426 and CCF-0545514.

## References

K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In *Proc. 17th International Conf. on Machine Learning*, pages 65–72. Morgan Kaufmann, San Francisco, CA, 2000.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

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

L. Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–849, 1998.

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

S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (Series B)*, 95:329–357, 2003.

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning*, pages 18–25, 2004.

P.K. Chan, W. Fan, A. Prodromidis, and S.J. Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems Journal*, November/December:67–74, 1999.

N.V. Chawla, L.O. Hall, K.W. Bowyer, and W.P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, 5: 421–451, 2004. ISSN 1533-7928.

A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *Advances in Neural Information Processing Systems*, 17, 2004.

A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000. ISSN 0885-6125.

W. Fan, H. Wang, and P.S. Yu. Mining extremely skewed trading anomalies. In *Proceedings of the 9th International Conference on Extending Database Technology*, pages 801–810, 2004.

U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41:174–211, 2001.

J.L. Fleiss. *Statistical Methods for Rates and Proportions*. John Wiley & Sons, 1981.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996. URL `citeseer.nj.nec.com/freund96experiments.html`.

M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite prgramming. *Journal of ACM*, 42: 1115–1145, 1995.

A.J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, pages 692–699, 1998.

Q. Han, Y. Ye, and J. Zhang. An improved rounding method and semidefinite programming relaxation for graph partition. *Mathematical Programming*, pages 509–535, 2002.

L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. ISSN 0162-8828.

S. Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1997. ISSN 0893-6080.

T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

Y. Kim, N.W. Street, and F. Menczer. Meta-evolutionary ensembles. In *IEEE International Joint Conference on Neural Networks*, pages 2791–2796, 2002.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5: 27–72, 2004.

D.D. Margineantu and T.G. Dietterich. Pruning adaptive boosting. In *14th International Conference on Machine Learning*, pages 211–218, 1997.

L. Mason, P. Bartlett, and J. Baxter. Direct optimization of margins improves generalization in combined classifier. *Advances in Neural Information Processing Systems*, 11:288–294, 1999.

A. McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI'99 Workshop on Text Learning*, 1999.

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, pages 169–198, 1999.

M.P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, pages 126–142. Chapman-Hall, 1993.

A. Prodromidis and P. Chan. Meta-learning in distributed data mining systems: Issues and approaches. In *Advances of Distributed Data Mining*. AAAI press, 2000.

A. Prodromidis and S. Stolfo. Pruning meta-classifiers in a distributed data mining system. In *Proc. of the First National Conference on New Information Technologies*, pages 151–160, Athens, Greece, October 1998.

R.J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Manteo, CA, 1993.

A.K. Rose and C. Engel. Currency unions and international integration. *Journal of Money, Credit and Banking*, 34(4):1067–89, 2002.

SDPLR, 2002. See the website: `http://dollar.biz.uiowa.edu/~sburer/software/SDPLR/`.

X. Shen, M. Boutell, J. Luo, and C. Brown. Multi-label machine learning and its application to semantic scene classification. In *International Symposium on Electronic Imaging*, San Jose, CA, January 2004.

W.N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 377–382, 2001.

H. Wang, W. Fan, P. Yun, and J. Han. Mining concept-drifting data streams using ensemble classifiers, 2003.

D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

G.U. Yule. On the association of attributes in statistics. *Philosophical Transactions of the Royal Society of London, Ser. A*, 194:257–319, 1900.

Z. Zhou, J. Wu, Y. Jiang, and S. Chen. Genetic algorithm based selective neural network ensemble. In *17th International Joint Conference on Artificial Intelligence*, pages 797–802, 2001.