DIMACS Technical Report 2009-21
November 2009

Tightened L_0-Relaxation Penalties for Classification[1]

by

Noam Goldberg,[2] Jonathan Eckstein[3]

[1]This material is based upon work funded in part by the U.S. Department of Homeland Security under Grant Award Number 2008-DN-077-ARI001-02. We thank Rob Schapire for helpful discussions. The first author would also like to thank Kristin Bennett for her comments during earlier stages of this work, and Martin Milanic for his comments regarding a related feature selection problem.

[2]Affiliated Graduate Student
RUTCOR, Rutgers University
640 Bartholomew Road
Piscataway NJ 08854; ngoldberg@rutcor.rutgers.edu
[3]Permanent Member
MSIS Department and RUTCOR, Rutgers University
640 Bartholomew Road
Piscataway NJ 08854; jeckstei@rci.rutgers.edu

ABSTRACT

In optimization-based classification model selection, for example when using linear programming formulations, a standard approach is to penalize the L_1 norm of some linear functional in order to select sparse models. Instead, we propose a novel integer linear program for sparse classifier selection, generalizing the minimum disagreement hyperplane problem whose complexity has been investigated in computational learning theory. Specifically, our mixed-integer problem is that of finding a separating hyperplane with minimum empirical error subject to an L_0 penalty. We show that common "soft margin" linear programming formulations for robust classification are equivalent to a continuous relaxation of our model. Since the initial continuous relaxation is weak, we suggest a tighter relaxation, using novel cutting planes, to better approximate the integer solution. We describe a boosting algorithm, based on linear programming with dynamic generation of cuts and columns, that solves our relaxation. We demonstrate the classification performance of our proposed algorithm with experimental results, and justify our selection of parameters using a minimum description length, compression interpretation of learning.

1 Introduction

Consider a binary classification problem with M training samples, each consisting of N real-valued attributes, represented as a matrix $A \in \mathbb{R}^{M \times N}$ whose rows correspond to observations and whose columns correspond to attributes. We are also given a vector of labels $y \in \{-1, 1\}^M$, defining a partition of the observations into a "positive" class $\Omega^+ = \{i \in \{1, \ldots, M\} \mid y_i = 1\}$ and a "negative" class $\Omega^- = \{1, \ldots, M\} \setminus \Omega^+$. We suppose we have potentially large set of base classifiers $h_u : \mathbb{R}^N \to \{-1, 0, 1\}$ indexed by the set $\mathcal{U} = \{1, \ldots, U\}$, and would like to train a weighted voting classifier

$$g(x) = \sum_{u \in \mathcal{U}} \lambda_u h_u(x),$$

for $0 \leq \lambda \in \mathbb{R}^U$. We classify any new observation $x \in \mathbb{R}^N$ as either positive or negative based on $\operatorname{sgn}(g(x))$.

The literature of learning algorithms for classification has considered various *loss functions* as classification performance measures. Common loss functions include the empirical 0/1 loss

$$\ell(y_i, g, A_i) = \mathbf{I}(yg(A_i) \leq 0), \tag{1}$$

where $\mathbf{I}(\cdot)$ is the 0/1 indicator function and A_i is the i$^{\text{th}}$ row of A, and the *soft margin* loss (with margin fixed to 1):

$$\ell(y_i, g, A_i) = (1 - yg(A_i))_+, \text{ where } (\cdot)_+ = \max\{\cdot, 0\}. \tag{2}$$

The *empirical risk minimization* strategy calls for minimization of the average loss (or equivalently the sum of losses) over the given data in order to determine λ. However, empirical risk minimization can result in overfitting and significantly larger losses with respect to (unseen) test data. Robust algorithms for classification mitigate this problem by considering other initial loss functions, regularizing, or adding a model complexity penalty. Everything else being the same, it seems desirable to follow Occam's principle and select models for which $\lambda_u > 0$ for the smallest possible subset of $\mathcal{U}$.

The L_0 norm of a vector is defined as its number of nonzero coefficients (and is thus not a true L_p norm). Optimally sparse classification models may be obtained by directly minimizing or penalizing the L_0 norm of λ. In order to avoid a hard combinatorial optimization problem, the authors of various methods such as LP-Boost, Lasso, and Support Vector Machines (SVMs) [17, 21, 16, 9] suggest using the L_1 or L_2 norms of λ instead of L_0. Provided one is using an appropriate loss function, such strategies have the computational advantage of involving only convex optimization problems. For example, minimizing the observation-wise sum of the loss function (2), plus an L_p penalty, for $p \geq 1$, yields the convex optimization problem

$$\min_{\lambda} \sum_{i=1}^{M} (1 - y_i g(A_i))_+ + \|\lambda\|_p \, .$$

Greedy algorithms have also been proposed for sparse feature and model selection [38, 43, 27].

The *Minimum Description Length* (MDL) approach attempts to balance the sparsity of a model against its accuracy by addressing the problem of model selection as a data compression optimization problem. The optimal model minimizes the sum of the size of an efficient encoding of the model and the size of encoding the observations which the model misclassifies. MDL is difficult to apply exactly in practice due to the difficulty of proving the efficiency of encoding schemes for a model. We assume that the encoded size of our model is linear with respect to the *support* of λ, the 0-1 vector with 1's in precisely the coordinates u for which $\lambda_u \neq 0$. This assumption allows us to take into account that some features may be more complex than others, and therefore may be assigned a greater complexity penalty. We find that the MDL principle and the compression interpretation are helpful in choosing the penalty parameters that otherwise might only be determined by experiment and cross-validation. Below, we derive and demonstrate the application of such penalty parameters for the case of Boolean monomial base classifiers.

The typical current methodology of robust voting classification methods is to allow for some outliers, observations which do not necessarily lie on the correct side of the hyperplane $g(x) = 0$, and to use regularization, penalizing either the L_1 or L_2 norms of λ. Robust linear separation formulations have been suggested by several authors such as Bennett and Mangasarian [10] and Cortes and Vapnik [16]. Cortes and Vapnik suggest maximizing a "soft margin" (4) in one of their papers first introducing Support Vector Machines (SVMs). SVM models find $g(\cdot)$ by maximizing the L_2 margin of separation on the space of features corresponding to $\mathcal{U}$. By "soft margin" it is meant that not all observations need to be separated and satisfy the same requirement of distance from the hyperplane. Graepel *et al.* [24] and Ratsch *et al.* [34] adapted the quadratic optimization formulation of SVMs using "soft margins" to a linear programming formulation.

In their LP-Boost algorithm, Demiriz, Bennett and Shawe-Taylor [17] use the following linear programming formulation based on the formulation of Graepel *et al.* [24] and Ratsch *et al.* [34]:

$$\begin{aligned}
\max \quad & \rho - D\sum_{i=1}^{M} \xi_i & & \text{(3a)}\\
\text{s.t.} \quad & y_i H_i \lambda + \xi_i \geq \rho & i = 1, ..., M \quad & \text{(3b)}\\
& \sum_{u=1}^{U} \lambda_u = 1 & & \text{(3c)}\\
& \xi_i, \lambda_u \geq 0 & i = 1, ..., M,\ u = 1, ..., U. \quad & \text{(3d)}
\end{aligned}$$

In this formulation, each observation $i \in \{1, ..., M\}$ has a variable ξ_i which allows it to have a margin smaller than ρ. The margin of observation i is equal to $y_i H_i \lambda$, where H_i is the i^{th} row of H, an $M \times U$ matrix whose elements are $h_{iu} = h_u(A_i)$. The parameter D penalizes the magnitude of each margin deviation $\xi_i = \rho - y_i H_i \lambda$; these deviations are illustrated in Figure 1. The matrix H, in which each column corresponds to a column of labels ($-1, 0, 1$ in our case) assigned by a base classifier $u \in \mathcal{U}$, usually has too many columns to be written in

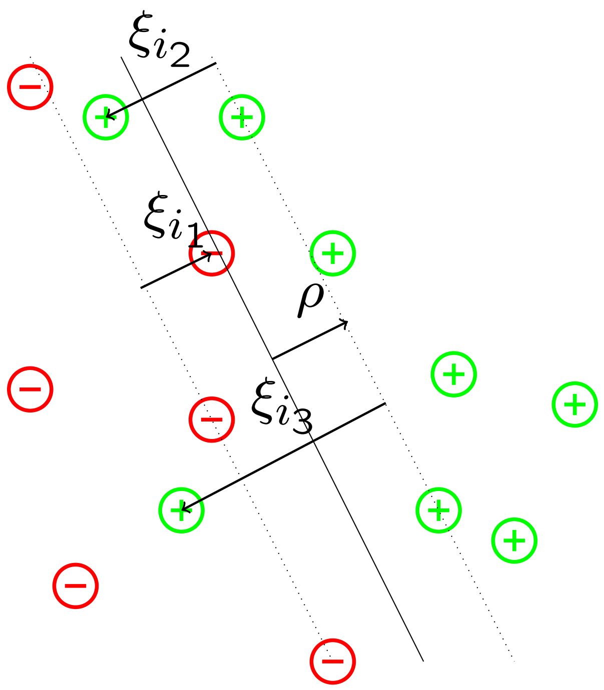


Figure 1: The margin ρ and margin deviations ξ_i illustrated in the "soft margin" LP formulation. The separating hyperplane is the solid line, and the maximum margin planes are the parallel dotted lines. Any point with $\xi_i > 0$ is called a *margin error*; points between the maximum margin lines are necessarily margin errors.

full as a part of the LP. Instead Demiriz *et al.* [17] propose a column generation procedure as a solution technique of the LP-Boost algorithm (to be exact, they in fact propose to generate cuts for the dual formulation).

In the formulation (3), the objective function is to maximize the L_1 margin of separation less a misclassification penalty. It has been shown by Bennett and Bredensteiner [9] that formulation (3) is equivalent to, for some appropriate choice of the constant D', the problem

$$\min \left\{ \sum_{u=1}^{U} \lambda_u + D' \sum_{i=1}^{M} \xi_i \;\middle|\; \operatorname{diag}(y) H\lambda + \xi \geq \mathbf{1},\ \text{and } \lambda, \xi \geq 0 \right\}. \tag{4}$$

The matrix H, in which each column corresponds to a column of labels ($-1, 0, 1$ in our case) assigned by a base classifier $u \in \mathcal{U}$, usually has too many columns to be represented explicitly, motivating the use of column generation procedures such as LP-Boost [17]. Column generation methods solve LPs with large numbers of variables by incrementally generating variables as they are needed; the generation of variables to append to the formulation involves a subsidiary optimization subproblem known as the *pricing* problem. In LP-Boost, the pricing problem coincides with the optimization problem to be solved by the *base learning* procedure, which is also known as the *weak learner* in the boosting literature [20, 32, 36]. Column generation has been used to solve large LPs since the early sixties, with a wide variety of applications starting with the seminal application of Gilmore and Gomory to the cutting stock problem [22, 30]. The method starts by solving a *restricted master problem* (RMP) that includes only a subset of the variables, and uses dual variable information to iteratively generate additional (primal) variables by solving a "pricing" subproblem. The algorithm appends the newly generated variables to the RMP and re-solves until no additional variables can be generated. The method is successful in practice when there are many more columns than rows, and the number of variables that need to be generated tends to be small.

2 Statistical learning theory justifications for our optimization formulation

We now motivate our approach by a brief appeal to statistical learning theory, which attempts to quantify prediction risk — the probability of error on the test (unseen) data for a given model. Freund and Schapire [20], drawing on work by Baum and Haussler [6] for neural networks, have bounded the prediction risk of weighted voting classifiers in terms of the L_0-norm of λ and the Vapnik-Chervonenkis (VC) dimension of the base classifiers; see for example [40]. Following later results by Bartlett [4], Schapire *et al.* [37], and Vapnik (see [38] section 5.5.6), algorithms for finding weighted voting classifiers that maximize the margin of separation, such as boosting and Support Vector Machines (SVMs), have been motivated by other risk bounds expressed in terms of the margin of separation, or equivalently the L_1 or L_2 norm of λ.

Luxburg, Bousquet, and Schölkopf have investigated the connection between statistical learning theory and compression for SVMs [42]. Although SVMs are designed to maximize the margin of separation in the space of features subject to a soft margin penalty, Luxburg *et al.* find that their compression-based bounds often perform better that margin-based bounds. Their analysis relates the idea of compression to the notion of separation margin by showing that the larger the L_2 margin of separation, the lower the precision needed to encode the voting weights λ, and the more the classifier description may be compressed.

While Luxburg *et al.* help to illuminate the relation between L_2 margin of separation and compression, their work raises the question whether there can be more direct approaches to "compressing" weighted voting classifiers. For example, are there efficient methods that more directly optimize sparsity and attempt to minimize the length of the classifier description?

Here, we adopt a strategy inspired by the MDL conception of learning: an imaginary sender and receiver are assumed to share the unlabeled data A and to agree in advance on the set of candidate features. The sender has access to the labels, and wishes to minimize the length of a two-part code intended to transmit them to the receiver: the first part of the code describes the classifier g, and the second part encodes which observations are misclassified by g. An "optimal" g minimizes the total length of this code.

MDL can be related to statistical learning theory through the compression interpretation of Vapnik [40], as well as that of Blum and Langford [11]. Vapnik gives a risk bound in terms of the compression coefficient in a scheme for encoding the labels of the training data [40]. Vapnik's bound requires that the set of classifier functions is finite and independent of the training data. Blum and Langford [11] avoid these conditions, but they require instead a code that can be used to encode both the training and test data labels. Let $\bar{L}(y, g)$ denote the length function of a code that can be used to encode the training and test data labels using g. The risk bound of Blum and Langford [11] applies also for infinite sets of functions $\mathcal{F}$, as long as the code of length $\bar{L}[y, g]$ can be used to encode the training labels y, without error, and is also able to encode an equal number of test labels $y' \in \{-1, 1\}^M$. Let us denote the corresponding observable attributes of the test data by $A' \in \mathbb{R}^{M' \times N}$. The resulting bound on the risk of $g \in \mathcal{F}$ holds with probability $1 - \delta$:

$$\tilde{R}[g] = \sum_{i=1}^{M'} \mathbf{I}(y'_i g(A'_i) \leq 0)/M' \leq \frac{\bar{L}[y, g]}{M} + \frac{\ln \delta}{M}. \tag{5}$$

Let $\dot{L}(g)$ denote the model code length of the weighted voting classifier g. We consider an upper bound on the code length that is a function of subsets of $\mathcal{U}$. Let $\mathcal{G}$ be the set of all weighted voting classifier models. Our objective then is to select a g that minimizes the sum of the model code length function (or an upper bound) $\dot{L}$ of g and the length of the misclassified data labels given g, that is,

$$\min_{g \in \mathcal{G}} \bar{L}(y, g) = \min_{g \in \mathcal{G}} \{\dot{L}(g) + L(y|g)\},$$

where $L(y|g)$ is the length of a code that encodes the labels of observations that are misclassified by g, requiring at most $\sum_{i=1}^M \mathbf{I}(y_i g(A_i)) \lceil \log M \rceil$ bits.

We further suppose that the base classifiers are partitioned into K subsets $\mathcal{U}_k$, $k = 1, \ldots, Q$, with each $\mathcal{U}_k$ representing the classifiers that have equal complexity or "risk"; this notion is related to the concept of *structural risk minimization* (SRM) [40, 39]. The corresponding weighted voting classifiers can then be decomposed into subsets $S_j = \text{conv}(\{h_u \mid u \in \bigcup_{k=0}^j \mathcal{U}_j\})$, for $j = 1, \ldots, K$, where $\text{conv}(U)$ denotes the convex hull of set U. Each of the sets $\mathcal{U}_k$ corresponds to one of K tables in a *code book* [40], present at both the sender and receiver, and an element of the k^{th} table can be identified using $\lceil \log |\mathcal{U}_k| \rceil$ bits.

To describe the classifier g, we must specify $\|\lambda\|_0$ such table indices, and also specify which table each element of $\{u \mid \lambda_u > 0\}$ corresponds to, the latter requiring at most $\|\lambda\|_0 \lceil \log K \rceil$ bits. Having thus identified the features, one must finally encode the numerical values of the separating hyperplane weights $\lambda \in [0, 1]^M$, which we show below requires at most $(\|\lambda\|_0 + 1)\lceil \log M \rceil$ bits to encode a set of affinely independent *support vectors*. The support vectors (as in SVM) are data points that lie on one of the hyperplanes given by $\sum_{u=1}^U \lambda_u h_u(x) = \pm 1$.

Theorem 2.1. *For a given set of points represented by the rows of $A \in \mathbb{R}^{M \times N}$, there is a two-part encoding of the hyperplane $g(x) = \lambda_u h_u(x) = 0$ that adds at most $(\|\lambda\|_0 + 1)\lceil \log M \rceil$ bits to the code length of $\{u \in \mathcal{U} \mid \lambda_u > 0\}$.*

Proof. Consider the $\|\lambda\|_0$-dimensional space defined by the features in $\{u \in \mathcal{U} \mid \lambda_u > 0\}$. The number of affinely independent vectors in this space can be at most $\|\lambda\|_0 + 1$. Thus at most $\|\lambda\|_0 + 1$ support vectors are needed to define one of the supporting hyperplanes $\sum_{u=1}^{U} \lambda_u h_u(x) = \pm 1$ as their affine combination. The parallel plane going through the origin is $\sum_{u=1}^{U} \lambda_u h_u(x) = 0$. For any two-part code where the sender and receiver share A_i for $i = 1, \ldots, M$, a data point vector can be identified using $\lceil \log M \rceil$ bits. $\square$

Using this information, a receiver can decode λ. The total length of the resulting encoded representation of g is:

$$\begin{aligned}
\dot{L}[g] &= \|\lambda\|_0 \lceil \log K \rceil + \sum_{u \in \mathcal{U}:\lambda_u > 0} \lceil \log \left|\mathcal{U}_{k(u)}\right| \rceil + (\|\lambda\|_0 + 1) \lceil \log M \rceil \\
&= \|\lambda\|_0 \left(\lceil \log K \rceil + (1 + 1/\|\lambda\|_0) \lceil \log M \rceil\right) + \sum_{u \in \mathcal{U}:\lambda_u > 0} \lceil \log \left|\mathcal{U}_{k(u)}\right| \rceil \qquad (6) \\
&\leq \|\lambda\|_0 \left(\lceil \log K \rceil + 2\lceil \log M \rceil\right) + \sum_{u \in \mathcal{U}:\lambda_u > 0} \lceil \log \left|\mathcal{U}_{k(u)}\right| \rceil \qquad (7) \\
&\leq \|\lambda\|_0 \left(\lceil \log K \rceil + 2\lceil \log M \rceil + \max_{u \in \mathcal{U}:\lambda_u > 0} \lceil \log \left|\mathcal{U}_{k(u)}\right| \rceil\right) \qquad (8)
\end{aligned}$$

where $k : \mathcal{U} \to \{1, \ldots, K\}$ and $\|\lambda\|_0 \geq 1$. Thus, by (8), minimizing $\|\lambda\|_0$ is equivalent to minimizing an upper bound on the code length $\dot{L}[g]$. Also, it follows from (7) that there is a tighter bound on the code length which can be written as a function that is linear in the support of λ.

3 The sparse minimum disagreement hyperplane problem

3.1 Problem formulation

The problem of finding a hyperplane g that minimizes the sum of (1) over $i = 1, \ldots, M$ is known as the *minimum disagreement halfspace* problem (MDH) [26, 1]. For the 0/1 loss (1), it turns out that even when a complete description of $\mathcal{U}$ is part of the input, the problem of finding a loss-minimizing hyperplane is $\mathcal{NP}$-hard [26, 7]. Mangasarian [31] and Bennett and Bredensteiner [8] have proposed heuristic approaches to this problem, based on nonlinear programming; both of these works refer to a hardness result appearing in the Ph.D. thesis of Heath [25]. The problem of minimizing the classification error is also known to be a case of *maximum feasible subsystem* problem: finding a feasible subsystem containing as many inequalities as possible from a given infeasible system $Ax \leq b$ [33].

We would like to extend the minimum disagreement hyperplane problem to penalize the L_0 norm of λ. We call the more general problem including an L_0 penalty the *sparse minimum disagreement hyperplane* problem (SMDH). In the basic version of the problem, we penalize all non-zero components of λ uniformly through the same penalty parameter C. The problem can be stated as:

Sparse Minimum Disagreement Hyperplane (SMDH)

Input: A matrix $H \in \{-1,0,1\}^{M\times U}$ of base classifier labels, a corresponding vector $y \in \{-1,1\}^M$ of sample labels, and a penalty $0 \le C < M$

Problem: To find a separating hyperplane, as specified by $\lambda \in \mathbb{R}^U_+$, such that $\sum_{i=1}^M \mathbf{I}(y_i H_i \lambda < 1) + C\left\|\lambda\right\|_0$ is minimized.

Note that we exclude the case that $C \ge M$ because it leads to the trivial solution $\lambda = 0$. We now formulate SMDH as a Mixed Integer Program (MIP), using binary variables μ_u to indicate whether feature u is used, and binary variables ξ_i to indicate whether observation i is misclassified:

$$\min_{\xi,\mu,\lambda}\left\{\sum_{i=1}^M \xi_i + C\sum_{u=1}^U \mu_u \;\middle|\; (\xi,\mu,\lambda) \in Q_{H,y} \cap \left(\{0,1\}^M \times \{0,1\}^U \times \mathbb{R}^U_+\right)\right\}, \tag{9}$$

where $Q_{H,y}$ is a "soft margin" classification polytope defined as

$$Q_{H,y} = \left\{(\xi,\mu,\lambda) \in [0,1]^M \times [0,1]^U \times \mathbb{R}^U_+ \;\middle|\; \begin{array}{l} \operatorname{diag}(y)H\lambda + (MK+1)\xi \ge \mathbb{1} \\ \lambda \le K\mu \end{array}\right\},$$

K being a suitably large constant and $\operatorname{diag}(y)$ the diagonal matrix with entries $y_1,\ldots,y_M$. We show that (9) is correct for all large enough K. Note that SMDH and formulation (9) always have a feasible solution. Therefore, since the objective value of any feasible solution to either SMDH or formulation (9) must be a nonnegative integer, each must always have an optimal solution. Thus, to prove the equivalence of (9) and SMDH, it is sufficient to show in the following theorem that the optimal solution values of SMDH and (9) are equal.

Theorem 3.1. *If $K \ge M^{M/2}$, then for any optimal solution (ξ^*,μ^*,λ^*) of (9), it must be that*

$$\sum_{i=1}^M \xi_i^* + C\sum_{u\in\mathcal{U}} \mu_u^* = \min_{\lambda\in\mathbb{R}^N_+}\sum_{i=1}^M \mathbf{I}(y_i H_i\lambda \le 0) + C\left\|\lambda\right\|_0 = \sum_{i=1}^M \mathbf{I}(y_i H_i \lambda^* \le 0) + C\left\|\lambda^*\right\|_0.$$

To prove this result, we will require the following two Lemmas.

Lemma 3.2. *If there exists a hyperplane $g(x) = \sum_{u\in\Gamma}\lambda_u x_u = 0$ that separates $S^+ \subseteq \Omega^+$ and $S^- \subseteq \Omega^-$, for some $\Gamma \subseteq \mathcal{U}$, then there exists λ^* such that $\lambda_u^* \le M^{M/2}$ for all $u \in \Gamma$, and $g(x) = \sum_{u\in\Gamma}\lambda_u^* x_u = 0$ also separates S^+ and S^-.*

Proof. Let $\hat{y}$ denote the subvector of y with elements $S^+ \cup S^-$ and $\hat{H}$ the submatrix of H with rows $S^+ \cup S^-$ and columns Γ. The points S^+ and S^- are linearly separable if the linear system $\operatorname{diag}(\hat{y})\hat{H}\lambda \ge \mathbb{1}$ has a feasible solution. The system of inequalities $\operatorname{diag}(\hat{y})\hat{H}\lambda \ge \mathbb{1}$ has a solution if it has a basic feasible solution λ^*. Let B denote a basis corresponding to a submatrix of $\operatorname{diag}(\hat{y})\hat{H}$. The basic feasible solution λ^* must satisfy the linear system

$$B\lambda^* = \mathbb{1}.$$

Now, by Cramer's rule and the non-negativity constraints, we have

$$\lambda_u^* = \frac{\det\left(B^{(u)}\right)}{\det(B)} \geq 0$$

where $B^{(u)}$ is the matrix B with column u replaced by $\mathbb{1}$. Since the rank of B is bounded by $|S^+ \cup S^-| \leq M$, we have using Hadamard's bound [13] that $\left|\det\left(B^{(u)}\right)\right| \leq M^{M/2}$. Since B is a basis, $\det(B) \neq 0$. Thus, by the definition of a determinant and since $B_{ij} \in \{-1, 0, 1\}$,

$$|\det(B)| = \left| \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} B_{i\sigma(i)} \right| \geq 1,$$

and the claim follows. □

Lemma 3.3. *If there exists a hyperplane $\sum_{t\in\Gamma} \lambda_t h_t(x) = 0$ that separates $S^+ \subseteq \Omega^+$ and $S^- \subseteq \Omega^-$, then there exists λ^* such that $\sum_{t\in\Gamma} \lambda_t^* h_t(x) = 0$ separates S^+ and S^- and $\|\lambda^*\|_0 \leq \operatorname{rank}(\tilde{H}) \leq |S^+| + |S^-|$, where $\tilde{H}$ is the submatrix of H with columns Γ and rows $S^+ \cup S^-$.*

Proof. Let

$$Y = \begin{bmatrix} y_{i_1} \tilde{H}_{i_1} \\ \vdots \\ y_{i_k} \tilde{H}_{i_k} \end{bmatrix},$$

where $\{i_1, \ldots, i_k\} = S^+ \cup S^-$. By feasibility of λ, the polyhedron

$$\left\{ \lambda \in \mathbb{R}_+^T \;\middle|\; \sum_{t\in\Gamma} y_i H_{it} \lambda_t \geq \mathbb{1} \text{ for } i \in S^+ \cup S^- \right\} \tag{10}$$

has a vertex corresponding to a basic feasible solution λ^*.

The number of nonzero components of each basic feasible solution is at most the rank of the constraint matrix, $\operatorname{rank}(Y)$, and $\operatorname{rank}(Y) = \operatorname{rank}(\tilde{H})$ since the rank of $\tilde{H}$ will not change if one simply negates some rows. Thus,

$$\|\lambda^*\|_0 = |\{t \in \Gamma \mid \lambda_t^* > 0\}| \leq \operatorname{rank}(Y) = \operatorname{rank}(\tilde{H}) \leq |S^+| + |S^-|. \qquad \square$$

Proof of Theorem 3.1. By the constraints $\lambda_u^* \leq K\mu_u^*$ of formulation (9),

$$\|\lambda^*\|_0 \leq \sum_{u\in\mathcal{U}} \mu_u^*.$$

By the constraint $\operatorname{diag}(y)H\lambda^* + (MK+1)\xi \geq \mathbb{1}$, it follows that $\sum_{i=1}^{M} \mathbf{I}(y_i H_i \lambda^* \leq 0) \leq \sum_{i=1}^{M} \xi_i^*$. Thus, by optimality of λ for SMDH, it follows that

$$\sum_{i=1}^{M} \mathbf{I}(y_i H_i \lambda \leq 0) + C\|\lambda\|_0 \leq \sum_{i=1}^{M} \mathbf{I}(y_i H_i \lambda^* \leq 0) + C\|\lambda^*\|_0 \leq \sum_{i=1}^{M} \xi_i^* + C \sum_{u\in\mathcal{U}} \mu_u^*. \tag{11}$$

We now prove the reverse inequality between the first and last quantities. The feasibility of λ for SMDH implies the linear separability of the subsets $S^+ = \{i \in \Omega^+ \mid \mathbf{I}(y_i H_i \lambda > 0)\}$ and $S^- = \{i \in \Omega^- \mid \mathbf{I}(y_i H_i \lambda > 0)\}$.

Now by Lemma 3.3, there exists λ' such that $\|\lambda'\|_0 \leq |S^+| + |S^-| \leq M$ and λ' separates S^+ and S^-. By the optimality of λ for SMDH, we also have $\|\lambda\|_0 \leq M$.

By Lemma 3.2 with $\Gamma = \{u \in \mathcal{U} \mid \lambda'_u > 0\}$, there exists λ'', with $\lambda''_u \leq M^{M/2} \leq K$ for all $u \in \mathcal{U}$, with the same support as λ', separating S^+ and S^-. Thus, $|y_i H_i \lambda''| \leq M^{M/2+1}$ for all $i = 1, \ldots, M$. Now let

$$\xi''_i = \begin{cases} 1 & \text{if } y_i H_i \lambda'' < 1 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mu''_u = \begin{cases} 1 & \text{if } \lambda''_u > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Then, for all $i \in \{1, \ldots, M\}$,

$$y_i H_i \lambda'' + \left(M^{M/2+1} + 1\right) \xi''_i \geq 1.$$

Thus, $(\xi'', \mu'', \lambda'')$ is feasible for (9), and $\sum_{i=1}^M \xi''_i = \sum_{i=1}^M \mathbf{I}(y_i H_i \lambda < 1)$, so, by the optimality of $(\xi^*, \mu^*, \lambda^*)$,

$$\sum_{i=1}^M \xi^*_i + C \sum_{u \in \mathcal{U}} \mu^*_u \leq \sum_{i=1}^M \xi''_i + C \sum_{u \in \mathcal{U}} \mu''_u \leq \sum_{i=1}^M \mathbf{I}(y_i H_i \lambda < 1) + C \|\lambda\|_0 .$$

Thus, all the relations in (11) hold with equality, and thus λ^* is also an optimal solution to SMDH. □

We next show that the continuous relaxation of (9), which can be stated as

$$\min \left\{ \sum_{i=1}^M \xi_i + C \sum_{u=1}^U \mu_u \mid (\xi, \mu, \lambda) \in Q_{H,y} \right\}, \tag{12}$$

is equivalent to the soft margin formulation (4) with appropriate choices of the penalties C and D. Formulation (4) is known to be equivalent to the soft margin maximization formulation (3) by the results of Ratsch *et al.* [35] and Bennett *et al.* [9], so that it follows that (3) is also equivalent to the relaxation of the SMDH problem relaxation (12). The theorem will also enable us to claim in Section 5 that our continuous relaxation formulation provides a tightened relaxation of the discrete SMDH formulation (9), by introducing novel cutting planes for the "soft margin" formulation.

Theorem 3.4. *(ξ, λ) is an optimal solution of (4) if and only if $(\xi/(MK+1), \lambda/K, \lambda)$ is an optimal solution of (12) with penalty $C = 1/(D(M + 1/K))$.*

Proof. First we show that every feasible solution (ξ, λ) of (4) corresponds to a feasible solution $(\xi/(MK+1), \lambda/K, \lambda)$ of (12) with objective value $\frac{1}{D(MK+1)}\left(\sum_{i=1} D\xi_i + \sum_{u=1}^{U} \lambda_u\right)$.

Assume (ξ, λ) is a feasible solution of (4). Now,

$$\mathrm{diag}(y)H\lambda + \xi = \mathrm{diag}(y)H\lambda + (MK+1)I\xi/(MK+1) \geq \mathbb{1}, \tag{13}$$

and $\mu = \lambda/K$ imply that $(\xi/(MK+1), \lambda/K, \lambda)$ is feasible for (12).

Letting $C = 1/(D(M+1/K))$, the objective value of the solution $(\xi/(MK+1), \lambda/K, \lambda)$ of (12) is

$$\sum_{i=1}^{M} \xi_i/(MK+1) + \frac{1}{D(M+1/K)} \sum_{u=1}^{U} \lambda_u/K = \frac{1}{D(MK+1)} \left(D\sum_{i=1}^{M} \xi_i + \sum_{u=1}^{U} \lambda_u \right). \tag{14}$$

Since the map $(\xi, \lambda, D) \mapsto (\xi/(MK+1), \lambda, 1/(D(M+1/K)))$ is a bijection (where $\mu = \lambda/K$ is fixed), and following the equality in (13) and (14), the converse also follows.

In (12), since the nonnegative variables μ_u have positive objective coefficients, each appears only in the constraint $\mu_u \geq \lambda_u/K$, and the objective is being minimized, all optimal solutions of (12) must satisfy $\mu = \lambda/K$. The claim then follows since the objective value of all feasible solutions under the bijective correspondence given by $(\xi, \lambda, D) \mapsto (\xi/(MK+1), \lambda, 1/(D(M+1/K)))$, where $\mu = \lambda/K$, is scaled by the constant $1/(D(MK+1))$. $\square$

The objective function of (9) minimizes misclassification plus a complexity penalty proportional to the number of features used. Thus, the SMDH formulation (9) corresponds to minimizing an upper bound on the total code length when the features being combined have equal complexity penalties. However, we may wish to assign different penalties c_u to different features $u \in \mathcal{U}$ (but most likely use the same c_u for all u in the same $\mathcal{U}_k$). Moreover, the SMDH formulation (9) may have many alternate solutions: even with respect to a fixed ξ and μ, there may be multiple feasible hyperplanes $\sum_{u\in\mathcal{U}:\mu_u=1} \lambda_u h_u(x) = 0$; we may wish to distinguish between these hyperplanes on the basis of the margin of separation $0 < \rho \leq 1$. We therefore generalize our formulation to include penalties that vary with u and to accept the margin ρ as a parameter; to make this parameterization sensible, we must normalize the weights λ, which requires an additional constraint. Finally, for reasons that should become clear in course, we give the formulation with respect to some subset of features $\Gamma \subseteq \mathcal{U}$:

$$\min \left\{ \sum_{i=1}^{M} \xi_i + \sum_{u\in\Gamma} c_u \mu_u \;\middle|\; (\xi, \mu, \lambda) \in Q_{H,y,\rho}(\Gamma) \cap \left(\{0,1\}^M \times \{0,1\}^{|\Gamma|} \times \mathbb{R}_+^{|\Gamma|} \right) \right\}, \tag{15}$$

where $Q_{H,y,\rho}(\cdot)$ is a soft margin classification polytope with a required margin ρ:

$$Q_{H,y,\rho}(\Gamma) = \left\{ (\xi, \mu, \lambda) \in [0,1]^M \times [0,1]^{|\Gamma|} \times \mathbb{R}_+^{|\Gamma|} \;\middle|\; \begin{array}{ll} \sum_{u\in\Gamma} y_i H_{iu}\lambda_u + (1+\rho)\xi_i \geq \rho & , i = 1, \ldots, M \\ \sum_{u\in\Gamma} \lambda_u = 1 & \lambda \leq \mu \end{array} \right\}.$$

3.2 Computational complexity

The SMDH problem generalizes the MDH problem, so that it is at least as hard to solve computationally. Specifically, the MDH problem is solved by (9) with $C = 0$. In the following we will refer to a solution (ξ, μ, λ) of (9), with $C = 0$, as an MDH solution. Höffgen, Simon and Van Horn [26] show that the MDH problem is not possible to approximate within a factor better than $(1 - \epsilon)\log M$ for any $\epsilon > 0$ unless $\mathcal{NP} \subseteq DTIME(M^{\log\log M})$, where $DTIME(n)$ is the class of problems that can be solved in deterministic time n. Arora, Babai, Stern and Sweedyk [1] improved the inapproximability factor to $2^{\log^{0.5-\epsilon} M}$, making the weaker assumption that $\mathcal{NP} \nsubseteq DTIME(M^{\mathrm{poly}(\log M)})$. By the restriction of SMDH with $C = 0$, the inapproximability result of Arora *et al.* directly applies. We state in the following theorem a more general result for SMDH with constant penalties, based on the inapproximability of MDH [1]. Note that if $C \geq M$, then SMDH has the trivial solution $\lambda = 0$ and $\xi_i = 1$ for all $i \in \{1, \ldots, M\}$. We will require the following Lemmas to derive our inapproximability result:

Lemma 3.5. *Given an MDH instance with input $H \in \{-1, 0, 1\}^{M\times U}$, $y \in \{-1, 1\}^M$, and some constant C, then for all $k \geq \lceil C\rceil M + 1$, there exists an $O(k\,\mathrm{poly}(M, U))$ reduction to an SMDH instance $H' \in \{-1, 0, 1\}^{Mk\times U}$ and $y' \in \{-1, 1\}^{Mk}$, such that MDH has an optimal solution $(\hat{\xi}, \hat{\mu}, \hat{\lambda})$ if and only if SMDH has an optimal solution $(\xi^*, \mu^*, \lambda^*)$, where $\sum_{i=1}^{Mk} \xi_i^* = k\sum_{i=1}^{M} \hat{\xi}_i$ and $\sum_{u=1}^{U} \mu_u^* \leq M$.*

Proof. Given the input (H, y) of MDH, and a constant C, construct an instance of SMDH (H', y'), by creating k duplicates of H_i in the matrix H', and k duplicates of y_i in the vector y', for each row H_i of H. Without loss of generality, assume that the rows of H' and y' are indexed such that $H_i = H'_i$ and $y_i = y'_i$ for $i = 1, \ldots, M$. Let $(\xi^*, \mu^*, \lambda^*)$ be an optimal SMDH solution for the input (H', y') with penalty C. Let $(\hat{\xi}, \hat{\mu}, \hat{\lambda})$ be an optimal solution of MDH, corresponding to formulation (9) for the input (H, y) with penalty $C = 0$, and value $z_{\mathrm{MDH}} = \sum_{i=1}^{M} \hat{\xi}_i + 0\sum_{u=1}^{U} \hat{\mu}_u = \sum_{i=1}^{M} \hat{\xi}_i$.
Now, since feasible solutions of MDH and SMDH must always exist, we only need to prove $z_{\mathrm{MDH}} = \sum_{i=1}^{M} \hat{\xi}_i = \sum_{i=1}^{Mk} \xi_i^*/k$. Assume to the contrary

$$z_{\mathrm{MDH}} \neq \sum_{i=1}^{Mk} \xi_i^*/k.$$

First, we note that we must have $z_{\mathrm{MDH}} \leq \sum_{i=1}^{Mk} \xi_i^*/k$, for otherwise $\sum_{i=1}^{M} \xi_i^* < z_{\mathrm{MDH}}$, which contradicts the optimality of z_{MDH}. On the other hand, if $z_{\mathrm{MDH}} < \sum_{i=1}^{Mk} \xi_i^*/k$, then

$$z_{\mathrm{MDH}} \leq \sum_{i=1}^{Mk} \xi_i^*/k - 1 \qquad (16)$$

By Lemma 3.3, since the sets $\{i \in \Omega^+ \mid \hat{\xi}_i = 0\}$ and $\{i \in \Omega^- \mid \hat{\xi}_i = 0\}$ are linearly separable, there exists λ corresponding to a hyperplane that separates $\{i \in \Omega^+ \mid \hat{\xi}_i = 0\}$

and $\{i \in \Omega^- \mid \hat{\xi}_i = 0\}$, with $\|\lambda\|_0 \leq M$. Let

$$\mu_u = \begin{cases} 1 & \text{if } \lambda_u \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Then the optimal SMDH solution value z_{SMDH} must satisfy

$$\begin{aligned} z_{\text{SMDH}} &= \sum_{i=1}^{Mk} \xi_i^* + C \sum_{u=1}^{U} \mu_u^* \\ &\leq k \sum_{i=1}^{M} \hat{\xi}_i + C \sum_{u=1}^{U} \mu_u && \text{[by optimality of SMDH]} \\ &\leq k z_{\text{MDH}} + CM \\ &< k\left(z_{\text{MDH}} + 1\right) && \text{[by } k \geq \lceil C \rceil M + 1\text{]} \\ &\leq \sum_{i=1}^{Mk} \xi_i^* && \text{[by (16)]} \\ &\leq z_{\text{SMDH}} \end{aligned}$$

From this contradiction, we conclude that $z_{\text{MDH}} = \sum_{i=1}^{Mk} \xi_i^*/k = \sum_{i=1}^{M} \xi_i^*$. □

Lemma 3.6. *An $\alpha(M)$ approximation factor for SMDH with constant penalty parameter C, for some $\alpha : \mathbb{N}_+ \to \mathbb{R}_+$ implies an $\alpha(M(\lceil C \rceil M + 1))\beta$ approximation of MDH for some $\beta \in O(1)$.*

Proof. We will reduce and MDH instance (H, y) to SMDH with a constant C using the reduction of Lemma 3.5, making $k = \lceil C \rceil M + 1$ duplicates of each row of H and y, in H' and y' respectively.

Suppose there is an $\alpha(kM) = \alpha(M(\lceil C \rceil M + 1))$-factor approximation algorithm for SMDH. Let (ξ, μ, λ) and $(\xi^*, \mu^*, \lambda^*)$ denote an SMDH solution of the approximation algorithm and optimal solution, respectively. Let $(\hat{\xi}, \hat{\mu}, \hat{\lambda})$ denote an optimal MDH solution, and z_{MDH} denote its value. Without loss of generality we may assume $\sum_{i=1}^{M} \hat{\xi}_i \geq \kappa$, for a constant $\kappa \in \mathbb{N}_+$. Otherwise, we can solve MDH exactly by excluding each subset up to size κ and finding the corresponding separating hyperplanes by linear programming in polynomial time. Now,

$$\begin{aligned} z_{MDH} = \sum_{i=1}^{M} \hat{\xi}_i &= \sum_{i=1}^{M(\lceil C \rceil M+1)} \xi_i^* / (\lceil C \rceil M + 1) && \text{[by Lemma 3.5]} \\ &\leq \left(\sum_{i=1}^{M(\lceil C \rceil M+1)} \xi_i^* + C \sum_{u=1}^{U} \mu_u^* \right) / (\lceil C \rceil M + 1) \\ &\leq \left(\sum_{i=1}^{M(\lceil C \rceil M+1)} \xi_i + C \sum_{u=1}^{U} \mu_u \right) / (\lceil C \rceil M + 1) \end{aligned}$$

$$\le \alpha(M(\lceil C\rceil M+1))\left(\sum_{i=1}^{M(\lceil C\rceil M+1)} \xi_i^* + C\sum_{u=1}^{U}\mu_u^*\right)/(\lceil C\rceil M+1) \quad \text{[by assumption]}$$

$$\le \alpha(M(\lceil C\rceil M+1))\left((\lceil C\rceil M+1)\sum_{i=1}^{M}\hat{\xi}_i + CM\right)/(\lceil C\rceil M+1) \quad \text{[by Lemma 3.5]}$$

$$\le \alpha(M(\lceil C\rceil M+1))(\sum_{i=1}^{M}\hat{\xi}_i + 1)$$

$$\le \alpha(M(\lceil C\rceil M+1))(1+1/\kappa)\sum_{i=1}^{M}\hat{\xi}_i,$$

which yields an $\alpha(M(\lceil C\rceil M+1))(1+1/\kappa) = \alpha(M(\lceil C\rceil M+1))\beta$-factor approximation for MDH, with $\beta \in O(1)$. □

Theorem 3.7. *The SMDH problem, with a constant C, cannot be approximated to within any constant factor, assuming $\mathcal{P} \neq \mathcal{NP}$.*

Proof. By Lemma 3.6 a constant factor approximation for SMDH yields a constant factor approximation for MDH and thus a contradiction [26, 1]. □

Theorem 3.8. *For any constant penalty C and $\epsilon > 0$, the SMDH problem cannot be approximated within a factor of $2^{\log^{0.5-\epsilon} M}$ unless $\mathcal{NP} \subseteq DTIME(M^{\text{poly}(\log M)})$.*

Proof. By Lemma 3.6, a $2^{\log^{0.5-\epsilon} M}$-factor approximation for SMDH, for some $\epsilon > 0$, yields a $\beta 2^{\log^{0.5-\epsilon}(M(\lceil C\rceil M+1))}$-factor approximation for MDH, for some $\beta \in O(1)$. Now, because $C \le M$,

$$\beta 2^{\log^{0.5-\epsilon}(M(\lceil C\rceil M+1))} \le \beta 2^{\log^{0.5-\epsilon} M^4} \beta 2^{4^{0.5-\epsilon}\log^{0.5-\epsilon} M} \le 2^{\log^{0.5-\epsilon'} M}$$

for some $0 < \epsilon' \le \epsilon$, $M_0 \in \mathbb{N}_+$ and all $M \ge M_0$. This is a contradiction with the inapproximability of MDH [1]. □

The continuous relaxations of models (9) and (15) can be quite weak. First we elaborate the on the weakness of the relaxation, and then we suggest novel cutting planes for the purpose of strengthening the relaxation.

4 Relaxing the hard problem and strengthening the relaxation

The weakness of the continuous relaxations of models (9) and (15) is reflected in their large *integrality gaps*. The integrality gap of our MIP formulation of SMDH may be defined as $\sup_{H,y} z(H,y)/z_R(H,y)$, where $z(H,y)$ and $z_R(H,y)$ are the optimal solution values of the SMDH MIP and its continuous relaxation, respectively; see Vazirani [41].

In order to show a lower bound for the integrality gap we will consider a particular construction of a simple SMDH instance with $C = 1$ and $\operatorname{diag}(y)H = I$ (the identity matrix), meaning that each base classifier covers only a single observation. Since each observation $i \in \{1, \ldots, M\}$ must be either classified correctly by the single classifier u with $y_i H_{iu} = 1$ and $\mu_u = 1$, or otherwise $\xi_i = 1$, this instance has an optimal integer solution of value M, where M of the μ_u and ξ_i variables assume a value of one and all of the remaining variables are zero. The relaxation of MDH, however, has the feasible solution $\xi_i = 1/(MK + 1)$ for $i = 1, \ldots, M$, and $\mu = 0$, with value $M/(MK + 1)$. In Lemma 3.2, we proved a large upper bound for the required constant K, *i.e.*, that formulation (9) is correct for all $K \geq M^{M/2}$. Smaller values of K that maintain the correctness of (9) may be possible. However, we can show a lower bound for any constant K that maintains the correctness of the formulation by constructing the following SMDH instance (different than the simple instance above used to demonstrate the gap): let

$$\operatorname{diag}(y)H = \begin{pmatrix} 1 & 0 & \ldots & & 0 \\ -1 & 1 & 0 & \ldots & 0 \\ 0 & -1 & \ddots & & 0 \\ & & -1 & 1 & 0 \\ -1 & \ldots & & -1 & 1 \end{pmatrix}.$$

Now, in order to admit the feasible SMDH solution $\lambda_1 = 1$, $\lambda_2 = 3$, $\ldots$,$\lambda_{M-1} = M - 1$, $\lambda_M = 0$, $\xi_1 = \ldots = \xi_{M-1} = 0$, $\xi_M = 1$ formulation (9) must have $K \geq M(M-1)/2$. Thus, the integrality gap of SMDH satisfies

$$\sup_{H,y} \frac{z(H,y)}{z_R(H,y)} \geq \frac{M}{M/(M(M(M-1)/2) + 1)} \geq M^2(M-1)/2.$$

Using the same simple SMDH instance, with $C = 1$ and $\operatorname{diag}(y)H = I$, we can also show a large lower bound factor of M for the MIP formulation (15). In the case of (15), due to the normalization constraint, the relaxation solution may have $\mu_u = \frac{1}{M}$ for all $u \in \mathcal{U}$, compared with the integer solution having $\mu_u = 1$ for all $u \in \mathcal{U}$. This instance, therefore, proves the integrality gap lower bound for (15), with

$$\frac{\sup_{H,y} z(H,y)}{z_R(H,y)} \geq \frac{\sum_{i=1}^{M} 1}{\sum_{i=1}^{M} 1/M} = M.$$

We now consider adding valid inequalities to (15) in order to strengthen its relaxation. We say that a base classifier h *distinguishes* between a pair (i, i') if it classifies them differently but classifies at least one of them correctly, *e.g.*, $h_u(A_i) = y_i \neq h_u(A_{i'})$. Let $S_{i,i'} = \{u \in \mathcal{U} \mid h_u(A_i) = y_i \neq h_u(A_{i'})\}$ denote the set of base classifiers that correctly classify observation i and distinguish it from i'. We consider the following inequality for each pair of observations $(i, i') \in (\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+)$:

$$\xi_i + \xi_{i'} + \sum_{u \in S_{i,i'} \cap \Gamma} \mu_u \geq 1. \tag{17}$$

Intuitively, such a cutting plane implies that either we misclassify at least one of the of the observations i or i', or we need to distinguish between the two using at least one of the distinguishing features in $S_{i,i'}$.

Theorem 4.1. *The inequalities* (17) *are valid, that is, they hold for all integer-feasible solutions of* (15).

Proof. Take any $(i,i') \in \Omega^+ \times \Omega^-$. If $\xi_i + \xi_{i'} \geq 1$ then (17) clearly holds. Otherwise, $i \in \Omega^+$ and $\xi_i = 0$ imply that $\sum_{t \in \Gamma} H_{it}\lambda_u \geq \rho$. Now, $\rho > 0$ implies that $h_u(A_i)\lambda_u > 0$ for some $t \in \Gamma$; $\lambda_u \geq 0$ and $h_u(A_i) = 1$ imply $\mu_u > 0$. The proof for $(i,i') \in \Omega^- \times \Omega^+$ is similar. $\square$

In the following, we will denote by $\mathcal{A}$ some subset of pairs in $(\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+)$. Now, we let

$$\mathcal{R}(\mathcal{A},\Gamma) = \left\{ (\xi,\mu,\lambda) \in [0,1]^M \times [0,1]^{|\Gamma|} \times \mathbb{R}_+^{|\Gamma|} \;\middle|\; \xi_i + \xi_{i'} + \textstyle\sum_{u \in S_{i,i'} \cap \Gamma} \mu_u \geq 1,\ \forall (i,i') \in \mathcal{A} \right\}$$

denote the polyhedron implied by the cutting planes (17) corresponding to the pairs of observations in $\mathcal{A}$.

As a direct consequence of Theorem 4.1,

$$Q_{H,y,\rho}(\Gamma) \cap \{0,1\}^M \times \{0,1\}^{|\Gamma|} \subseteq Q_{H,y,\rho}(\Gamma) \cap \mathcal{R}(\mathcal{A},\Gamma) \subseteq Q_{H,y,\rho}(\Gamma)$$

Similarly, it can be shown by letting $\rho = 1$ that the inequalities (17) are valid for (9), so that

$$Q_{H,y} \cap \{0,1\}^M \times \{0,1\}^U \subseteq Q_{H,y} \cap \mathcal{R}(\mathcal{A},\mathcal{U}) \subseteq Q_{H,y}.$$

Finally, we note that following work done on characterization of the set cover polytope [3, 15], or equivalently by applying a special case of Chvatal-Gomory cuts, the inequalities (17) can be further strengthened by introducing certain inequalities derived from triples of observation pairs. However, we have not pursued this line of research further.

5 L_0-Relaxed Boosting: a boosting formulation with relaxed L_0 complexity penalties

We now describe a boosting algorithm for the continuous relaxation of SMDH, strengthened by inequalities of the form (17). Our algorithmic approach is similar to Demiriz *et al.* [17]: We use column generation to iteratively generate the columns of $\mathcal{U}$ as needed. At each iteration, the set of features is restricted to the subset $\Gamma \subseteq \mathcal{U}$ of the columns that have been generated so far. At each iteration, the base learner algorithm generates a new feature from

$\mathcal{U} \setminus \Gamma$. Given a current set of features Γ and set of pairs $\mathcal{A}$, we arrive at the relaxation

$$\begin{aligned}
\min_{\lambda,\mu,\xi} \quad & \sum_{i=1}^{M} \xi_i + \sum_{u=1}^{U} c_u \mu_u && && \text{(18a)} \\
\text{s.t.:} \quad & \sum_{u \in \Gamma} y_i H_{iu} \lambda_u + (1+\rho)\xi_i \geq \rho && i = 1, ..., M && \text{(18b)} \\
& \sum_{u \in \Gamma} \lambda_u = 1 && && \text{(18c)} \\
& \xi_i + \xi_{i'} + \sum_{u \in S_{ii'} \cap \Gamma} \mu_u \geq 1 && (i,i') \in \mathcal{A} && \text{(18d)} \\
& \mu_u - \lambda_u \geq 0 && u \in \Gamma && \text{(18e)} \\
& \lambda_u, \mu_u, \xi_i \geq 0 && u \in \Gamma,\ i = 1, ..., M && \text{(18f)}
\end{aligned}$$

5.1 Dual formulation, base learning problem, and termination

We next derive the dual formulation of (18) in order to formulate the pricing/base learning problem. The dual formulation is also useful in proving the algorithm's termination condition, which guarantees an optimal solution, or an approximately optimal solution. Let w_i, α, $v_{ii'}$, and q_u correspond respectively to the dual variables (or Lagrange multipliers) of i^{th} constraint (18b), (18c), constraint (18d) of pair (i,i'), and the u^{th} constraint (18e).

$$\begin{aligned}
\max_{w,v,\alpha,q} \quad & \sum_{i=1}^{M} \rho w_i + \sum_{(i,i') \in \mathcal{A}} v_{ii'} + \alpha && && \text{(19a)} \\
\text{s.t.:} \quad & && && \\
& \sum_{(k,l): i \in \{k,l\}} v_{kl} + (1+\rho) w_i \leq 1 && i \in \{1, \ldots, M\} && \text{(19b)} \\
& \sum_{\substack{(k,l) \in \mathcal{A}: \\ k=i \vee l=i}} v_{ii'} + q_u \leq c_u && u \in \Gamma && \text{(19c)} \\
& \alpha + \sum_{i=1}^{M} y_i \hat{H}_{it} w_i - q_u \leq 0 && u \in \Gamma && \text{(19d)} \\
& w \geq 0, q_u, v_{ii'} \geq 0 && (i,i') \in \mathcal{A}, u \in \Gamma && \text{(19e)}
\end{aligned}$$

Substituting $q_u = c_u - \sum_{(i,i') \in \mathcal{A}: S_{ii'} \ni t} v_{ii'}$, which is that largest value of q_u that satisfies (19c), will preserve all feasible solutions, since choosing q_u as large as possible must satisfy the corresponding inequality (19d), for any given feasible (α, w), and q_u does not appear in the objective or any constraint other than (19c) or (19d). Thus, we can state the dual LP of (18) as

$$\begin{aligned}
\max_{u,v,\alpha} \quad & \sum_{i=1}^{M} \rho w_i + \sum_{(i,i')\in\mathcal{A}} v_{ii'} + \alpha && && \text{(20a)}\\
\text{s.t.:} \quad & \sum_{\substack{(k,l)\in\mathcal{A}:\\ k=i\vee l=i}} v_{kl} + (1+\rho)w_i \leq 1 && i \in \{1,\dots,M\} && \text{(20b)}\\
& \alpha + \sum_{(i,i')\in\mathcal{A}:S_{i,i'}\ni u} v_{ii'} + \sum_{i=1}^{M} y_i H_{iu} w_i \leq c_u && u \in \Gamma && \text{(20c)}\\
& w \geq 0, v_{ii'} \geq 0 && (i,i') \in \mathcal{A}. && \text{(20d)}
\end{aligned}$$

At each iteration, boosting algorithms invoke a base learning algorithm to find the best hypothesis to add to the classifier. The base learning problem that arises with our new formulation (18) involves finding $u \in \mathcal{U}$ that produces the most violated constraint of the form (20c), and is similar to the base learning problem of LP-Boost, but involves an additional weight measure $v : (\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+) \to \mathbb{R}_+$ corresponding to the Lagrange multipliers of the constraints (18d); it takes the form

$$\bar{c} = \min_{u\in\mathcal{U}} \left\{ c_u - \sum_{i=1}^{M} y_i H_{iu} w_i - \sum_{(i,i')\in\mathcal{A}:u\in S_{ii'}} v_{ii'} - \alpha \right\}, \tag{21}$$

The minimizer of $\bar{c}$ can be determined irrespective of the constant α, which may be moved outside the "min" operation. In the general case, it is unsurprising that the problem (21) is $\mathcal{NP}$-hard. This fact is confirmed in Section 6 in the special case where the columns of H correspond to vertices of subcubes of the binary hypercube. Whenever $\bar{c} \geq 0$, the dual constraints (20c) must be satisfied for all $u \in \mathcal{U}$, and the dual LP (20) becomes feasible. Thus, $\bar{c} \geq 0$ proves optimality of a selection of base classifiers (and columns) $\Gamma \subseteq \mathcal{U}$, and defines our termination condition. It is also possible to terminate early while guaranteeing an approximate solution of the relaxation [30], although in the experiments of Section 7 we did not find this necessary.

Let us denote the optimal solution value of the SMDH problem (15) by $z(\Gamma)$, and its relaxation (18) optimal value by $z_R(\Gamma)$ (for a particular fixed instance (H, y, ρ)). Having solved the relaxation (18) for a set of columns $\Gamma \subseteq \mathcal{U}$, it may be possible terminate the column generation procedure early while guaranteeing a lower bound on the solution value. The following lower bound is based on a general lower bound commonly used for early termination of column generation [30].

Suppose b is an upper bound on the number of features in the optimal solution, *i.e.*, $\sum_{u=1}^{U} \mu_u^* \leq b$ for the optimal solution $(\xi^*, \mu^*, \lambda^*)$ of (18), with value $z_R(\Gamma)$. By Lemma 3.3, a trivial choice is $b = M$, but it may be possible to find smaller values in terms of M and c, for example if $c_u = C \geq 1$ for all $u \in \mathcal{U}$, then $b = \lceil M/C \rceil$. If $\bar{c}$ is the optimal reduced cost computed by (21), then the optimal solution value can improve by at most $|\bar{c}b|$, so that the following bounds are guaranteed for the solution of $z_R(\mathcal{U})$:

$$z_R(\Gamma) + \bar{c}b \le z_R(\mathcal{U}) \le z_R(\Gamma). \tag{22}$$

Further, for the integer optimal solution of the master problem $z(\mathcal{U})$, we can derive a tighter lower bound when c_u are integer for all $u \in \mathcal{U}$, namely

$$\lceil z_R(\Gamma) + \bar{c}b \rceil \le \lceil z_R(\mathcal{U}) \rceil \le z(\mathcal{U}) \le \hat{z}, \tag{23}$$

where $\hat{z}$ is any upper bound on $z(\mathcal{U})$. The upper bound $\hat{z}$ can be computed by simply rounding up any non-integer variables in the solution (ξ, μ, λ) of (18), or by any other rounding that maintains feasibility for (18). Note that we can rewrite (15), whenever the coefficients c_u are rational, as an equivalent optimization problem with integer coefficients c_u.

Finally, we note that in order to compute a risk upper bound such as (5), we can use any feasible rounding of an intermediate solution (ξ, μ, λ), with value $\hat{u}$, as an upper bound for $z(\mathcal{U})$ and thus for the code length $\min_{g \in \mathcal{G}} \bar{L}[y, g]$.

Algorithm 1 L_0-RBoost

```
1: Input: $M \times N$ matrix $A$ and labels $y \in \{-1,1\}^M$
2: Output: $(\xi, \mu, \lambda)$
3: Let $\Gamma \leftarrow \{1,2\}$, where $h_1(A_i) = 1$ and $h_2(A_i) = -1$ for all $i \in \{1,\dots,M\}$, $\mathcal{A} \leftarrow \emptyset$
4: repeat
5:    Solve (18) and obtain the solution $(\xi, \mu, \lambda)$ and Lagrange multipliers $(w, v, \alpha)$
6:    Solve the base learning problem:
      $$u^* = \operatorname{argmin}_{u \in \mathcal{U}} c_u - \sum_{(i,i') \in \mathcal{A}: S_{ii'} \ni u} v_{ii'} - \sum_{i=1}^{M} y_i H_{iu} w_i - \alpha$$
      $$\bar{c} = c_{u^*} - \sum_{(i,i') \in \mathcal{A}: S_{ii'} \ni u^*} v_{ii'} - \sum_{i=1}^{M} y_i H_{iu^*} w_i - \alpha$$
7:    $\Gamma \leftarrow \Gamma \cup \{u^*\}$
8:    $\mathcal{A} \leftarrow \mathcal{A} \cup \{(i,i') \in (\Omega + \times \Omega^-) \cup (\Omega^- \times \Omega^+) \mid h_{u^*}(A_i) \ne h_{u^*}(A_{i'})\}$
9:    Let $V = \left\{(i,i') \;\middle|\; \xi_i + \xi_{i'} + \sum_{u \in S_{i,i'} \cap \Gamma} \mu_u < 1\right\}$
10:   if $\bar{c} \ge 0$ and $V \ne \emptyset$ then
11:      $\mathcal{A} \leftarrow \mathcal{A} \cup V$
12:   end if
13: until $\bar{c} \ge 0$ and $V = \emptyset$
```

5.2 The boosting algorithm

Algorithm 1 is our proposed L_0-RBoost algorithm. We generate columns for the primal formulation (18), rather than (equivalently) generating cuts for a dual formulation as suggested by Demiriz *et al.* for LP-Boost. An attractive property of the dual formulation is the ease of finding an initial feasible solution by assigning the observations equal weights $w_i = 1/M$. In our case, however, we are interested in the integrality of the solution vectors ξ and μ, so we prefer to work in the primal space. We initialize Γ to contain two columns that are both "simple" and easy to compute, corresponding to the constant base classifiers: we take $\Gamma = \{1, 2\}$, where $h_1(A_i) = 1$ and $h_2(A_i) = -1$ for all $i \in \{1, ..., M\}$. Next, we iterate by solving the relaxation (18), and then solving a base learning problem (21) to find the best pair of variables λ_u and μ_u to be added. Note that it may be beneficial in practice to add more than a single column u in each iteration, though we did not experiment with such strategies. We terminate when the dual becomes feasible, *i.e.*, when (20c) is satisfied for all $u \in \mathcal{U}$, implying an optimal solution.

If the number of observations is not too large, we may simply solve (18) using all possible cuts, that is, start with $\mathcal{A} = (\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+)$. The number of possible cutting planes (17) is $2|\Omega^+||\Omega^-|$, which is polynomial in M, but may be prohibitively large for some datasets. In order to handle large input datasets, we dynamically generate the cutting planes; for each newly added column u, we add the cutting planes (17) that correspond to pairs of positive and negative observations that are distinguished by u, that is, pairs in the set

$$\big\{(i, i') \in (\Omega + \times \Omega^-) \cup (\Omega^- \times \Omega^+) \;\big|\; h_u(A_i) \neq h_u(A_{i'})\big\} \setminus \mathcal{A}. \tag{24}$$

These cutting planes are designed to push up the value of the newly generated variable μ_u as close as possible to 1; they might otherwise be as small as λ_u. Note that, in the interest of speed, we may choose to add some heuristically determined subset of the possible cutting planes corresponding to (24), rather than all of them. One approach we have found beneficial uses a similarity measure between the observation vectors A_i and $A_{i'}$; we will elaborate on this idea in Section 7. Finally, before terminating, we make sure to add all remaining violated cuts in step 11 of Algorithm 1. This step prevents premature termination, and is especially effective in increasing the value of the variables ξ_i for any remaining misclassified observations. Initially, the number of misclassified observations is large, but it drops as the algorithm progresses; delaying step 11 thus allows fewer cuts to be generated.

5.3 Analysis and margin maximization with L_0 relaxation penalties

Algorithm 1 solves a tighter relaxation of the (generalized) SMDH problem than the straightforward relaxation that minimizes the L_1-norm of λ (*i.e.*, as used by LP-Boost). Alternatively, using any polynomial time algorithm to solve the LP in step 5, since the number of cuts (18d) is at most $|\Omega^+||\Omega^-|$, the running time of Algorithm 1 can be shown to be polynomial in the dimensions M and U, and the size of the encoding of the coefficients ρ and c [29].

A disadvantage of our formulation may be that we may not know how to set the required margin ρ. An alternative may be to try to maximize the margin within the optimization problem. Let $z(\rho)$ denote the optimal solution value of (18) with parameter ρ. The following formulation maximizes ρ subject to a (relaxed) code length penalty proportional to $z(\rho)$:

$$\max_{\xi,\mu,\lambda,\rho} \left\{ \rho - \beta \left(\sum_{i=1}^{M} \xi_i + D \sum_{u=1}^{U} \mu_u \right) \;\middle|\; (\xi, \mu, \lambda) \in Q_{H,y,\rho}(\Gamma) \cap \mathcal{R}(\mathcal{A}, \Gamma) \right\} \tag{25}$$

Making ρ a variable on both the left and right-hand side of the constraint $y_i H_i \lambda + (1+\rho)\xi_i \geq \rho$ results in a quadratic optimization problem. Alternatively, one may fix the coefficient $1+\rho$ on the left-hand side to be a large upper bounding constant such as $2 \geq 1+\rho$. However, this results in a poorer relaxation of the integer solution. We proceed to show that (25) can be solved in polynomial time in the input (when the input includes $\mathcal{U}$).

Lemma 5.1. *Suppose (ξ, μ, λ) is a feasible solution of (18) with $\rho = \rho' > 0$. Then there exists $\lambda^* \geq 0$ such that (ξ, μ, λ^*) is feasible for (18), with $\rho = \rho'' \geq M^{-(M/2+1)}$.*

Proof. Let $\Gamma = \{u \in \mathcal{U} \mid \lambda_u > 0\}$. The feasibility of (ξ, μ, λ) for (15) implies the linear separability of $S^+ = \{i \in \Omega^+ \mid \xi_i = 0\}$ and $S^- = \{i \in \Omega^- \mid \xi_i = 0\}$. By Lemmas 3.2 and 3.3, there exists a $\lambda' \geq 0$ that separates S^+ and S^-, satisfies $\lambda'_u \leq M^{M/2}$ for all $u \in \mathcal{U}$, and has $\|\lambda'\|_0 \leq M$. We then have

$$\sum_{u\in\Gamma} y_i H_{iu} \lambda'_u + \left(\sum_{u=1}^{U} \lambda'_u + 1 \right) \xi_i \geq 1,$$

for all $i \in \{1, \ldots, M\}$.

Dividing by $\sum_{u\in\mathcal{U}} \lambda'_u$ and substituting the variable $\lambda^*_u = \frac{\lambda'_u}{\sum_{u\in\mathcal{U}} \lambda'_u}$, we have $\sum_{u\in\mathcal{U}} \lambda^*_u = 1$ and

$$\sum_{u\in\Gamma} y_i H_{iu} \lambda^*_u + \frac{\sum_{u\in\mathcal{U}} \lambda'_u + 1}{\sum_{u\in\mathcal{U}} \lambda'_u} \xi_i \geq 1 / \sum_{u\in\mathcal{U}} \lambda'_u.$$

Now, letting $\rho'' = 1/\sum_{u\in\mathcal{U}} \lambda'_u$,

$$\sum_{u\in\Gamma} y_i H_{iu} \lambda^*_u + (1+\rho'')\xi_i \geq \rho''.$$

where $\rho'' = 1/\sum_{u=1}^{U} \lambda'_u \geq M^{-(M/2+1)}$, for all $i \in \{1, \ldots, M\}$. Thus, (ξ, μ, λ^*) is feasible for (18), with $\rho = \rho''$. □

Lemma 5.2. *The optimal solution value of (18), $z(\rho)$, is a convex function of ρ.*

Proof. Consider the solutions (ξ', μ', λ') with $\rho = \rho_1$ and $(\xi'', \mu'', \lambda'')$ with $\rho = \rho_2$ to (18). Then, by adding α times the i^{th} constraint (18b) of Q_{H,y,ρ_1} and $(1-\alpha)$ times constraint (18b) of Q_{H,y,ρ_1} for $\alpha \in [0,1]$,

$$\begin{aligned} &\alpha y_i \hat{H}_i \lambda' + \alpha(1+\rho_1)\xi'_i + (1-\alpha) y_i \hat{H}_i \lambda'' + (1-\alpha)(1+\rho_2)\xi''_i \\ &\qquad = y_i \hat{H}_i (\alpha\lambda' + (1-\alpha)\lambda'') + (1 + \alpha\rho_1 + (1-\alpha)\rho_2)\, \xi_i \geq \alpha\rho_1 + (1-\alpha)\rho_2, \end{aligned}$$

and thus constraint (18b) is feasible for

$$(\bar{\xi}, \bar{\mu}, \bar{\lambda}) = (\alpha\xi' + (1-\alpha)\xi'', \alpha\mu' + (1-\alpha)\mu'', \alpha\lambda' + (1-\alpha)\lambda'')$$

in (18) with $\rho = \alpha\rho_1 + (1-\alpha)\rho_2$. Clearly, the constraints (18c), (18d) and (18e) are also feasible for $(\bar{\xi}, \bar{\mu}, \bar{\lambda})$. We then have:

$$\begin{aligned} z\left(\alpha\rho_1 + (1-\alpha)\rho_2\right) &\leq \sum_{i=1}^{M} \bar{\xi}_i + \sum_{u=1}^{U} c_u \bar{\mu}_u \\ &= \sum_{i=1}^{M} \left(\alpha\xi_i' + (1-\alpha)\xi_i''\right) + \sum_{u=1}^{U} c_u \left(\alpha\mu_u' + (1-\alpha)\mu_u''\right) \\ &= \alpha z(\rho_1) + (1-\alpha) z(\rho_2) \end{aligned}$$

□

Theorem 5.3. *Maximizing the L_1 margin subject to a relaxed L_0 penalty, as specified by* (25)*, can be approximated to within a factor of $1-\epsilon$, for any $\epsilon > 0$, in time polynomial in the input size M, U, the encoding size of the coefficients c and ρ, and $1/\epsilon$.*

Proof. By Lemma 5.2 the optimal solution of (18), $z(\rho)$, is a convex function of ρ. Thus, $\rho - \beta z(\rho)$ is a concave function of ρ, for any $\beta \geq 0$. By Lemma 5.1, it suffices to consider ρ within the interval $[1/M^{M/2+1}, 1]$. We consider approximately maximizing $\rho - \beta z(\rho)$ by binary search.

The number of evaluations needed in order to approximate the optimal margin ρ^* within $1-\epsilon$, for any given $\epsilon > 0$, is $\log\left(\frac{1-M^{-M/2-1}}{\epsilon}\right)$. Each such evaluation can be done by solving a linear program, which is solvable in time polynomial in M, T, and the encoding size of c and ρ. □

6 Application using Boolean monomial base classifiers

Now, assuming that our data matrix A is binary, we consider a specific class of base classifiers corresponding to Boolean monomials. Note that any given data matrix $A \in \mathbb{R}^{M \times N'}$ can be binarized using a number of binary attributes that is at most polynomially larger in M and N' than N' [12, 23]. A *monomial* on $\{0,1\}^N$ is simply a function $m : \{0,1\}^N \to \{0,1\}$ of the form

$$m_{J,C}(x) = \prod_{j \in J} x_j \prod_{c \in C} (1 - x_c), \tag{26}$$

where J and C are disjoint subsets of $\{1, \ldots, N\}$. To each monomial (J, C), we associate two features: a positive feature u^+ and a negative feature u^-; the corresponding base classifiers are $h_{u^+(J,C)}(x) = m_{J,C}(x)$ and $h_{u^-(J,C)} = -m_{J,C}(x)$. Boolean monomials also correspond to subcubes of the binary hypercube. A monomial $m_{J,C}$ is said to *cover* (or contain) a vector $x \in \{0,1\}^N$ if $m_{J,C}(x) = 1$. We define the *coverage* of a monomial $m_{J,C}$ to be

$\mathrm{Cover}(J, C) = \{i \mid m_{J,C}(A_i) = 1\}$. Let the positive reward associated with a monomial (J, C) be

$$f^+(J, C) = w(\Omega^+ \cap \mathrm{Cover}(J, C)) - w(\Omega^- \cap \mathrm{Cover}(J, C)) \quad + \sum_{\substack{i \in \Omega^+ \cap \mathrm{Cover}(J,C) \\ i' \in \Omega^- \setminus \mathrm{Cover}(J,C)}} v_{ii'},$$

and the negative reward associated with (J, C) be

$$f^-(J, C) = w(\Omega^- \cap \mathrm{Cover}(J, C)) - w(\Omega^+ \cap \mathrm{Cover}(J, C)) \quad + \sum_{\substack{i \in \Omega^- \cap \mathrm{Cover}(J,C) \\ i' \in \Omega^+ \setminus \mathrm{Cover}(J,C)}} v_{ii'},$$

where $w(S) = \sum_{i \in S} w_i$. The base learning problem is then to find (J, C) such that

$$f(J, C) = \max\left\{f^+(J, C), f^-(J, C)\right\} - c(|J| + |C|) \tag{27}$$

is maximized, where $c(k)$ denotes the complexity penalty of a monomial of order k. Here, we focus on base classifiers that correspond to Boolean monomials. The problem of finding a monomial of arbitrary order that maximizes $f(J, C)$ is a generalization of the maximum monomial agreement problem [28, 18], and thus it trivially follows by the restriction $v_{ii'} = 0$ for all (i, i') that the problem is $\mathcal{NP}$-hard. However, in many learning applications, it is reasonable to bound the order of the monomials by a small constant. Clearly, if the order of monomials is bounded by a constant K, then the monomial that maximally agrees with the data can be found in polynomial time by simple enumeration.

The MDL/compression approach discussed in Section 2 suggests one way of determining $c(k)$: the code book contains K tables, and the table for monomials of order k contains $|\mathcal{U}_k| = 2^k\binom{N}{k}$ entries. Based on the code length upper bound (7), normalizing by the approximate $\log M$ cost of a misclassified observation, we may define the cost of a Boolean monomial of order $k = |J| + |C|$ as:

$$c(k) = \frac{\log\left(2^k\binom{N}{k}\right) + \log K}{\log M} + \kappa = \frac{k + \log\binom{N}{k} + \log K}{\log M} + \kappa, \tag{28}$$

where κ is a constant that accounts for the bits that encode the weights λ. We can approximate the additional cost associated with the term $(\|\lambda\|_0 + 1)/\|\lambda\|_0$ in (7), which is associated with the cost of encoding λ, by choosing $\kappa \in [1 + 1/M, 2]$.

7 Experimental work and discussion

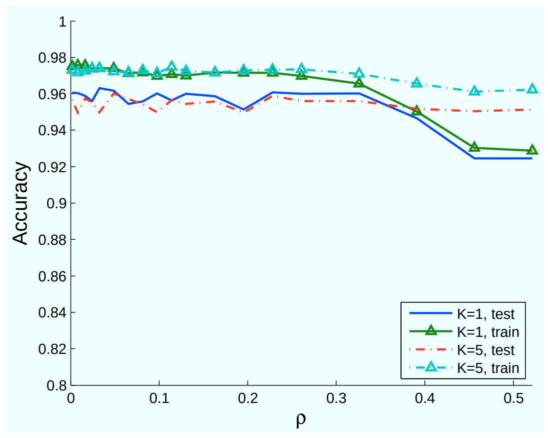
We compare the classification performance of L_0-RBoost (Algorithm 1) with LP-Boost with a Boolean monomial base classifier. The formulation used in LP-Boost is the ν-LP formulation (3), where $D = \frac{1}{\nu M}$ [17, 35]. The parameter ν also corresponds to an upper bound on the fraction of margin errors, *i.e.*, $|\{i \mid \xi_i > 0\}|/M$. In our experiments, we consider base learning problems comprising either of all Boolean monomials of order $k = 1$, or all monomials up to order $k = 5$. Although the order of monomials that we consider is bounded by constant, so that the maximum monomial agreement can be solved in polynomial time, it may be too computationally intensive to enumerate all monomials up to $k = 5$. Therefore, in order to find the monomial that best agrees with the data, we adapted our earlier branch-and-bound algorithm for maximum monomial agreement [19]; see also [23].

In our implementation of Algorithm 1, we have found it beneficial to add only a subset of the possible cuts in step 8 of the algorithm. From our experiments with binary data matrices A, we have found a small Hamming distance between the vectors A_i and $A_{i'}$ to be a good indication of the potential of the corresponding cut to tighten the relaxation. Specifically, for fast processing of the cuts, after adding a base classifier u^*, we scan in quadratic time the pairs $(i, i') \in \Omega^+ \times \Omega^-$ for which $h_{u^*}(A_i) \neq h_{u^*}(A_{i'})$ and add only those cuts corresponding to pairs whose Hamming distance is less than a fixed factor of the minimum Hamming distance found.

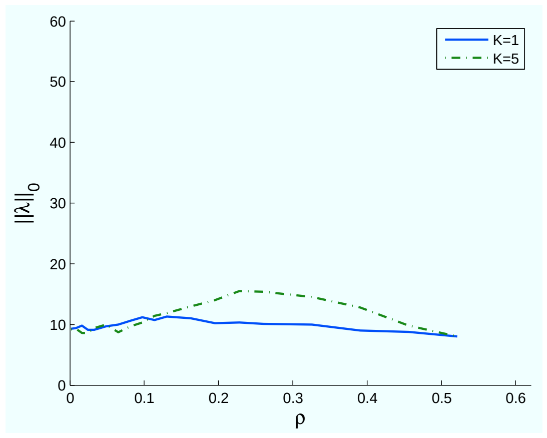
That we are better able to minimize $\|\lambda\|_0$ than the solution of (3), for a given margin of separation, is suggested by Theorem 4.1. The theorem implies that adding the valid inequalities (17) should strengthen the "soft margin" formulations (4) and (3), considering them as continuous relaxations of (9). The LP formulation (3) is known to find sparse classifiers, but is also sensitive to the choice of the tunable penalty parameter D. By assigning a small enough penalty D (large ν) in (3), it is possible to find a "degenerate" classifier with large ρ by assigning many observations to be outliers.

In our experiments with L_0-RBoost, we fixed the parameters c_u of formulation (18) to equal the code length penalty of Boolean monomial base classifiers (28). The approximating constant κ in (28) was set to 1.5. We then investigated the dependence of L_0-RBoost and LP-Boost on the tunable parameters ρ and ν, with respect to classification performance and sparsity.

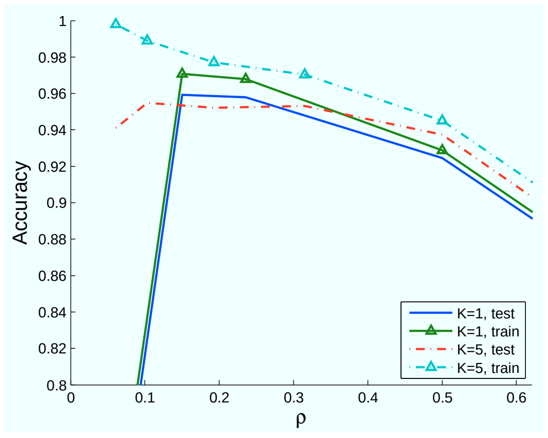
Figures 2-6 display the the training accuracy, testing accuracy, and $\|\lambda\|_0$ attained by L_0-RBoost and LP-Boost, in relation to the algorithms' tunable parameters, for monomials of degree $K = 1$ or up to $K = 5$. Each point of the plots corresponds to 10 replications of 10-fold experiment. k-fold experiments involve a partitioning of the dataset into 10 parts, each of which is used as a test set in an experiment, while the remaining 9/10 of the data is used to train a classifier. The classification performance results are then averaged over the total 100 experiments. We can see that for both algorithms, the models selected tend to overfit the training data for small value of the parameters. The overfitting is most apparent with LP-Boost and $K = 5$. The experiments show that, over the entire range of parameter values, L_0-RBoost generalizes well compared with LP-Boost. We also find that L_0-RBoost is robust with respect to a wide range of choices of the parameter ρ. Specifically, even with very small values of ρ, we obtain classification models that generalize well. In LP-Boost, the performance of the algorithm is highly sensitive to the choice of the parameter ν. When the input data is not linearly separable by the set of base classifiers $\mathcal{U}$, as is typically the case when $\mathcal{U}$ is the set of monomial classifiers with $K = 1$, a constant base classifier with zero margin (classifying all to be in a single class) becomes optimal. Further, the value of ν that may be considered too small varies for the different datasets; in Figure 6, the performance



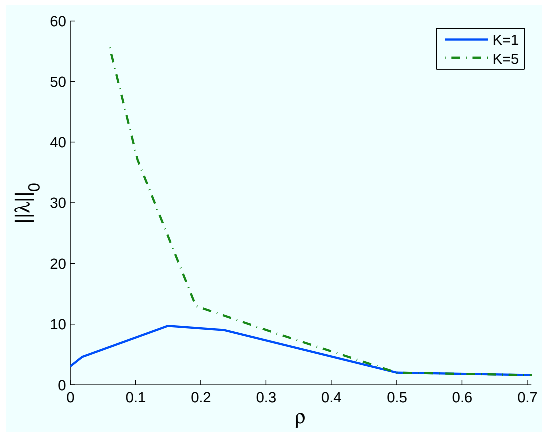
(a) L_0-RBoost accuracy vs. margin



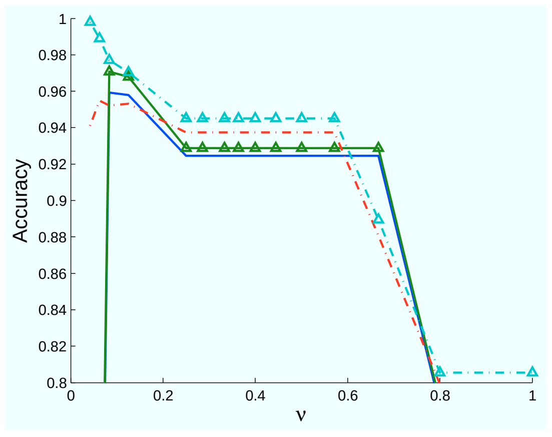
(b) L_0-RBoost sparsity vs. margin



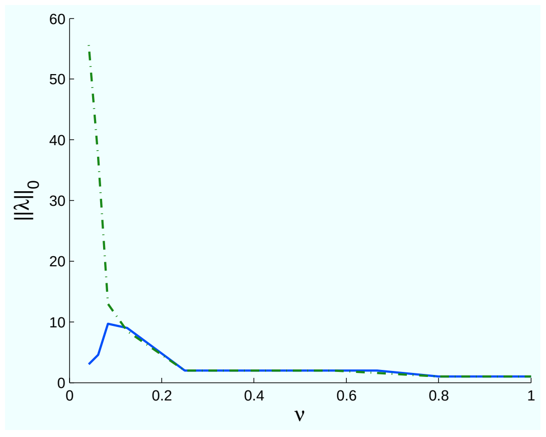
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

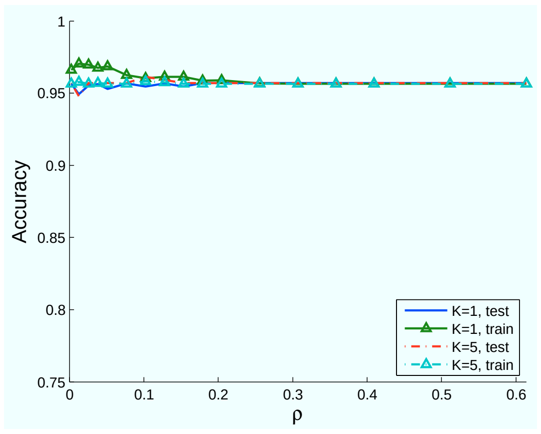


(e) LP-Boost accuracy vs. ν

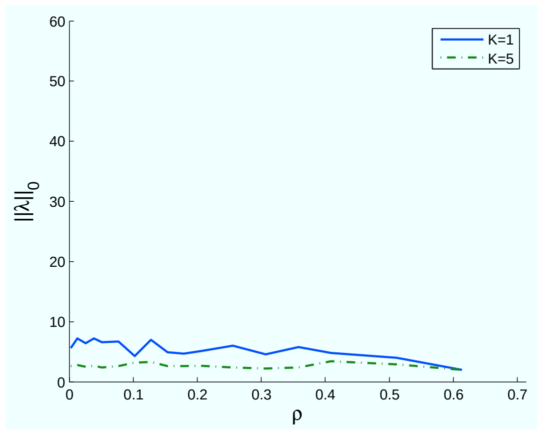


(f) LP-Boost sparsity vs. ν

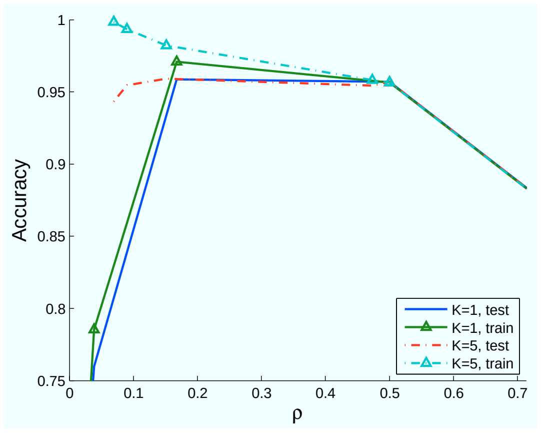
Figure 2: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the BCW dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



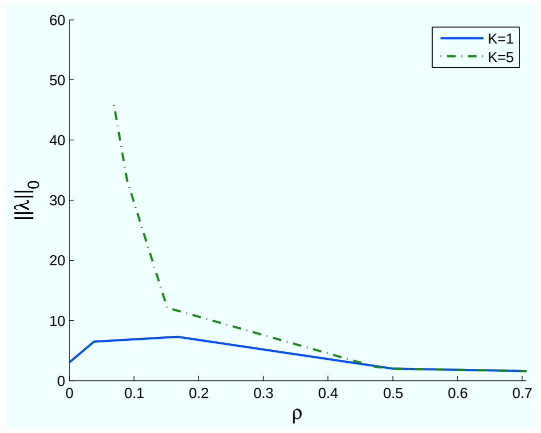
(a) L_0-RBoost accuracy vs. margin



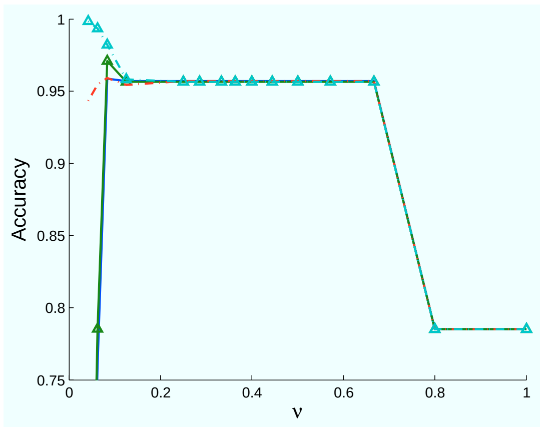
(b) L_0-RBoost sparsity vs. margin



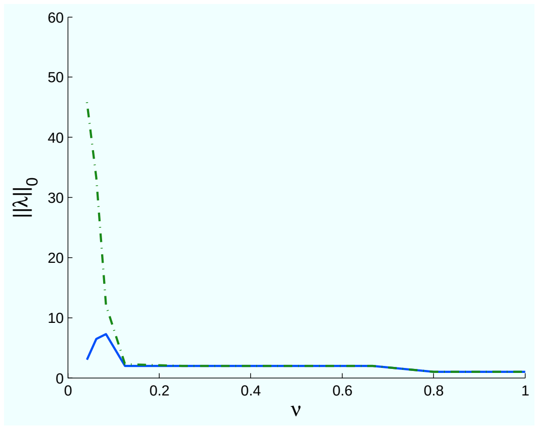
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

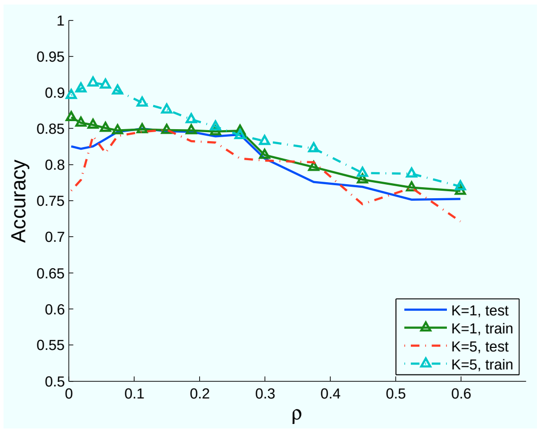


(e) LP-Boost accuracy vs. ν

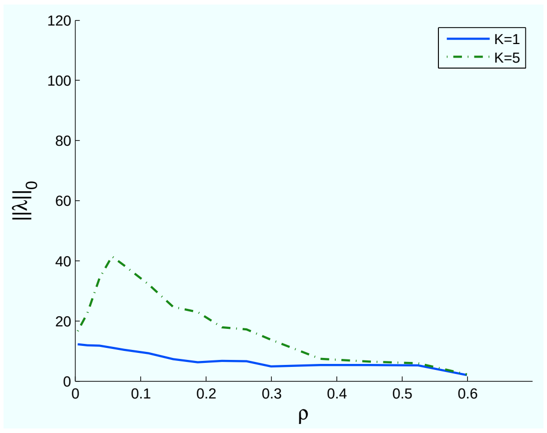


(f) LP-Boost sparsity vs. ν

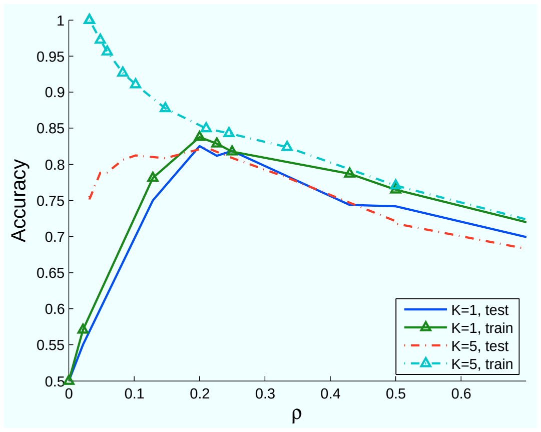
Figure 3: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the VOTE dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



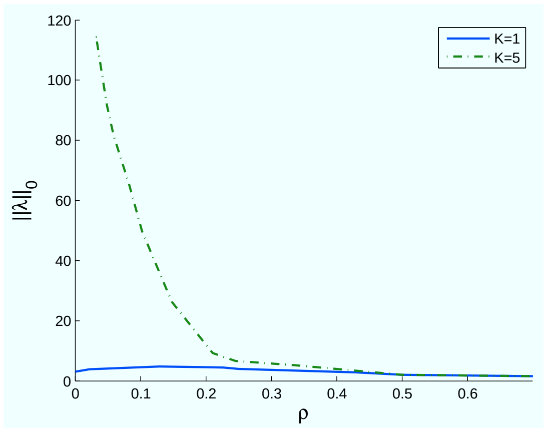
(a) L_0-RBoost accuracy vs. margin



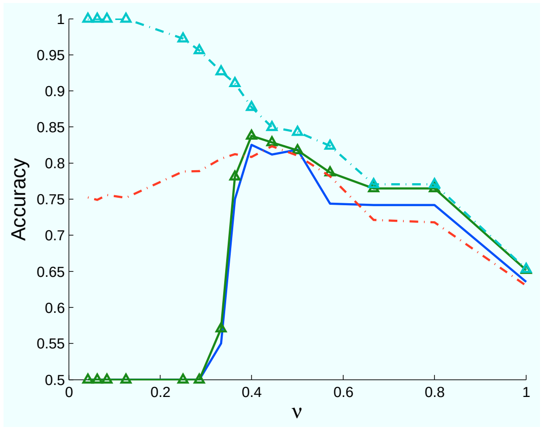
(b) L_0-RBoost sparsity vs. margin



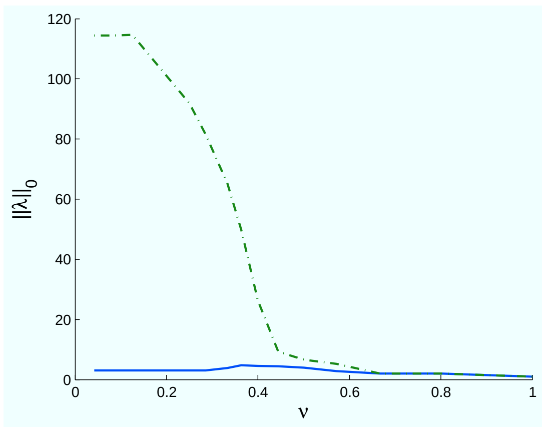
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

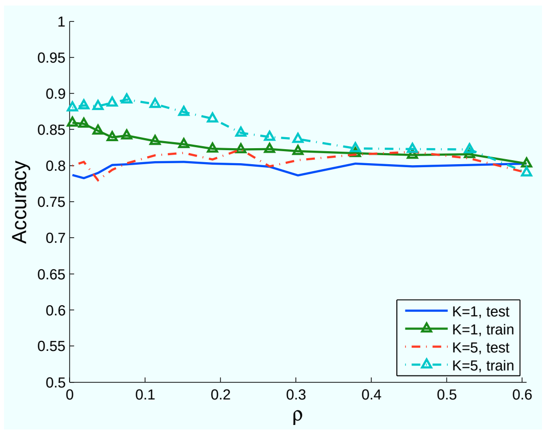


(e) LP-Boost accuracy vs. ν

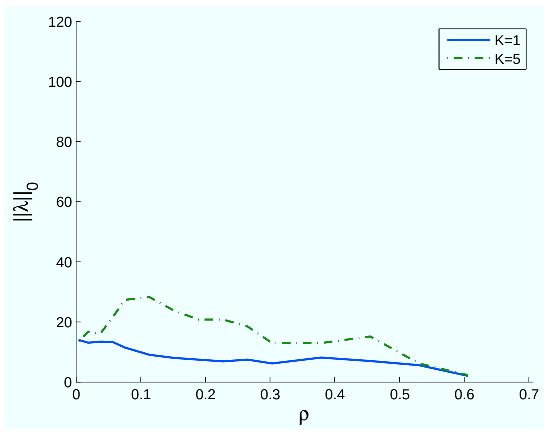


(f) LP-Boost sparsity vs. ν

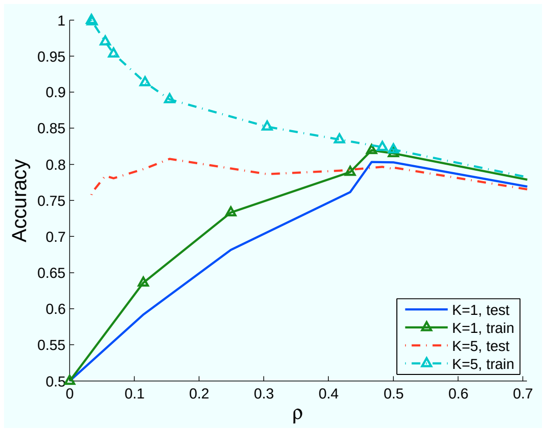
Figure 4: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the CLVHEART dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



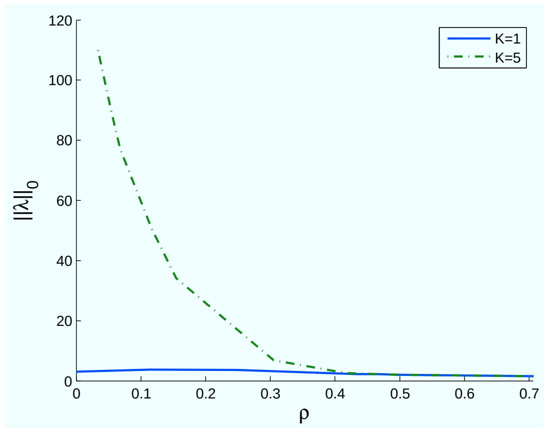
(a) L_0-RBoost accuracy vs. margin



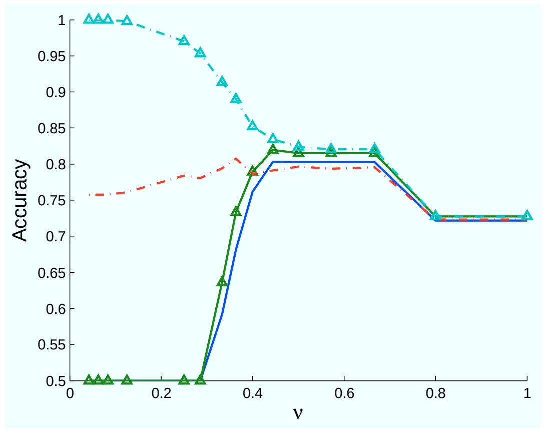
(b) L_0-RBoost sparsity vs. margin



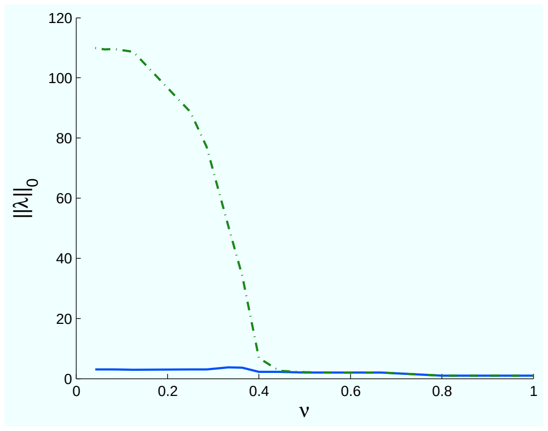
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

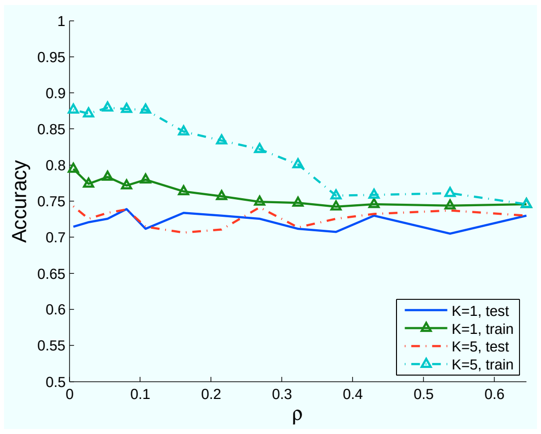


(e) LP-Boost accuracy vs. ν

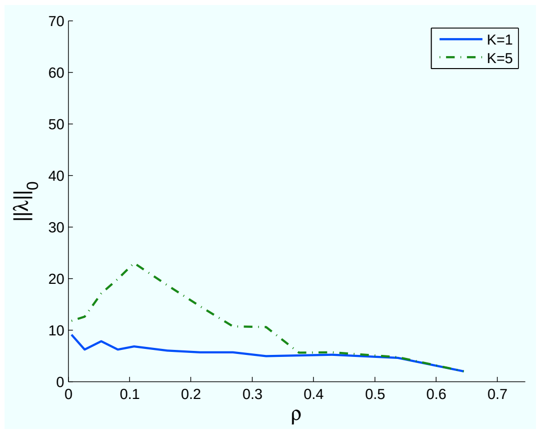


(f) LP-Boost sparsity vs. ν

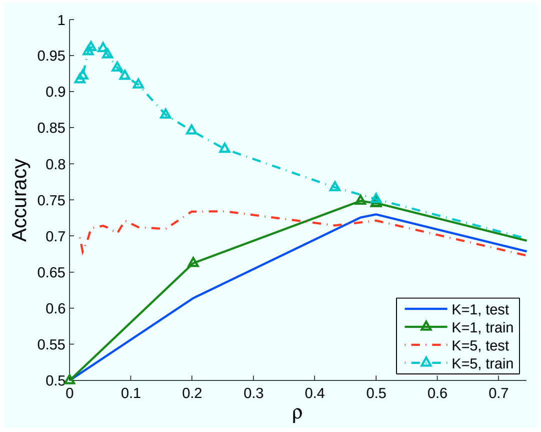
Figure 5: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the HUHEART dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



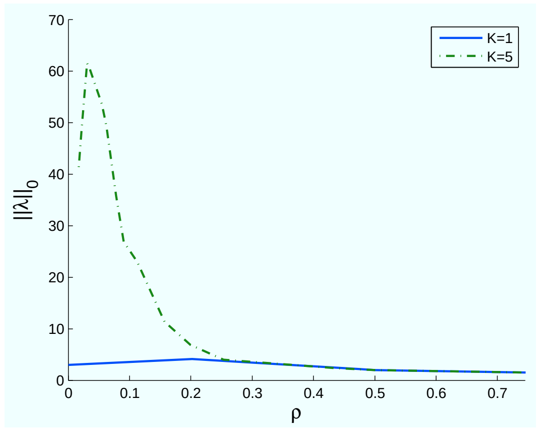
(a) L_0-RBoost accuracy vs. margin



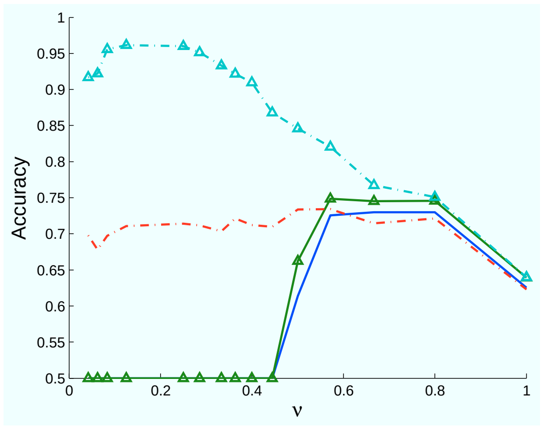
(b) L_0-RBoost sparsity vs. margin



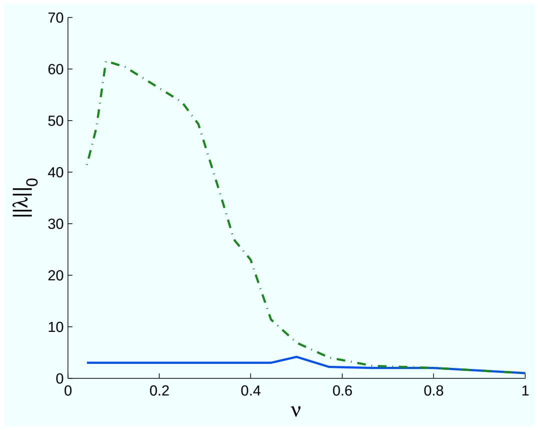
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

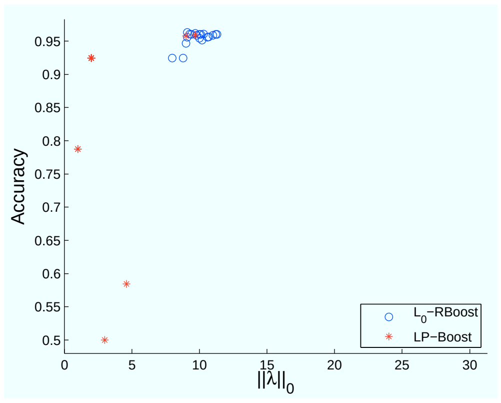


(e) LP-Boost accuracy vs. ν

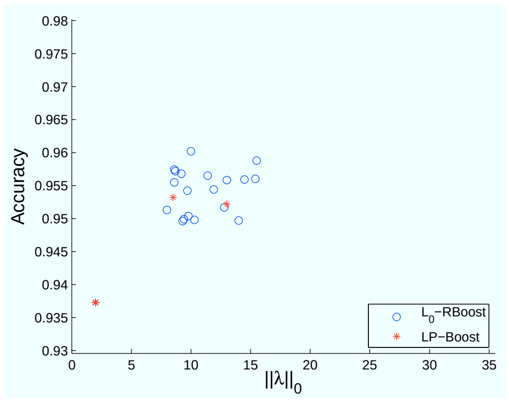


(f) LP-Boost sparsity vs ν

Figure 6: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers up to order $K = 1$ and $K = 5$ for the SONAR dataset. Each point of the plot is computed by averaging the accuracies of a 10-replication, 10-fold experiment.

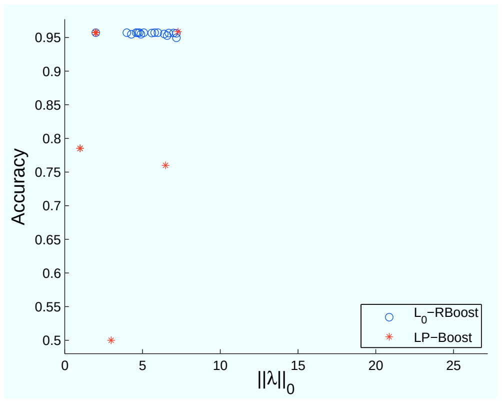


(a) $K = 1$

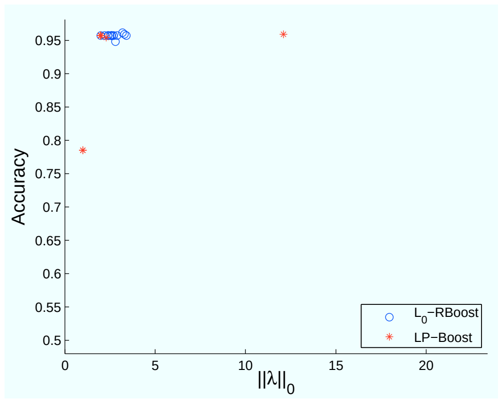


(b) $K = 5$

Figure 7: Test accuracy vs. $\|\lambda\|_0$ on the BCW dataset



(a) $K = 1$



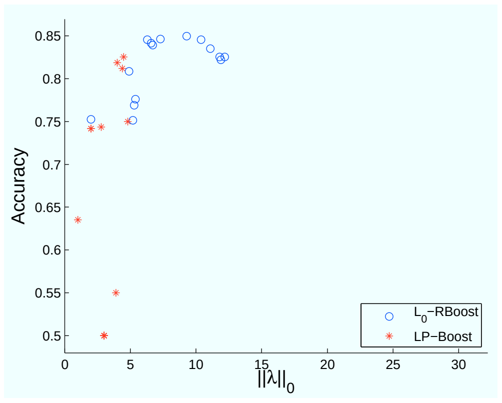
(b) $K = 5$

Figure 8: Test accuracy vs. $\|\lambda\|_0$ on the VOTE dataset

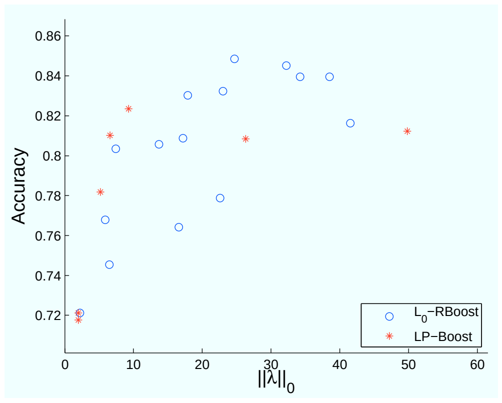
of LP-Boost with $K = 1$ is poor or mediocre for $\nu \leq 0.5$, while in Figure 4 we can see that the classification performance of LP-Boost on the CLHEART dataset, for $K = 1$, peaks at $\nu \approx 0.4$.

Figures 7-11 plot accuracy on the test set versus $\|\lambda\|_0$, summarizing the classification performance of the experiments depicted in Figures 2-6. As in Figures 2-6, each point corresponds to an average over ten replications of a 10-fold experiment. In general, the $\|\lambda\|_0$ values produced by L_0-RBoost are bounded within a smaller interval, which is to be expected, given that the cost coefficients c_u and κ are fixed in all of the experiments. However, it is apparent that the classifiers computed by L_0-RBoost are more accurate than LP-Boost's for all but a few values of $\|\lambda\|_0$.

Table 1 shows the results of experiment using the fixed parameter value $\rho = 20/M$ for five binarized UCI datasets [2]. The top part of the table compares L_0-RBoost with monomial base classifiers of maximum degree $K = 1$ or $K = 5$ to LP-Boost using the same base

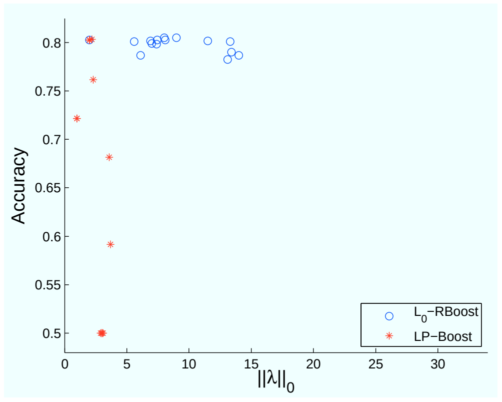


(a) $K = 1$

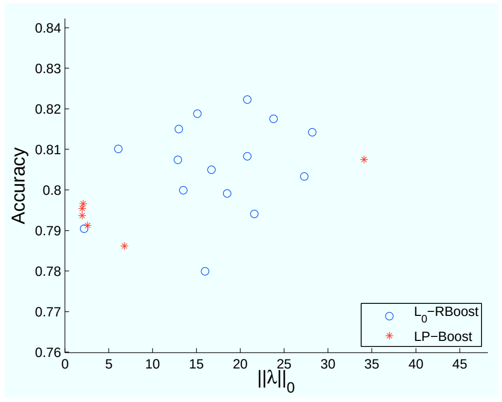


(b) $K = 5$

Figure 9: Test accuracy vs. $\|\lambda\|_0$ on the CLVHEART dataset

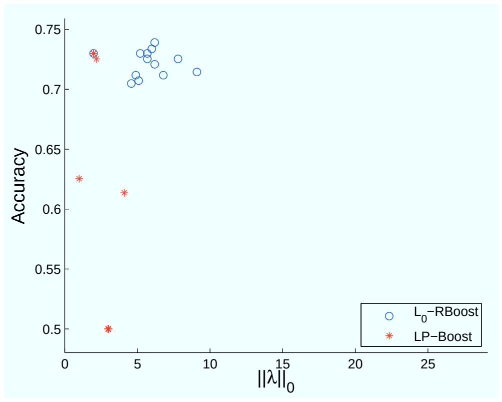


(a) $K = 1$

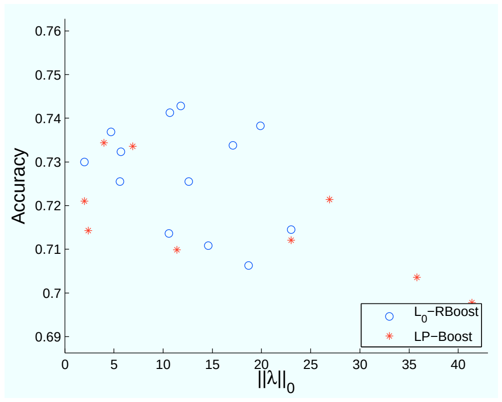


(b) $K = 5$

Figure 10: Test accuracy vs. $\|\lambda\|_0$ on the HUHEART dataset



(a) $K = 1$



(b) $K = 5$

Figure 11: Test accuracy vs. $\|\lambda\|_0$ on the SONAR dataset

Method	Datasets									
	BCW		VOTE		CLVHEART		HUHEART		SONAR	
	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$
L_0RBoost $K = 1$	0.963	9.1	0.950	6.0	0.846	10.3	0.812	10.4	0.712	7.2
L_0RBoost $K = 5$	0.950	9.4	0.960	3.0	0.833	38.9	0.807	27.4	0.725	21.3
LPBoost $K = 1$	0.925	3.8	0.957	2	0.774	7.6	0.803	3.8	0.735	8.6
LPBoost $K = 5$	0.937	5.5	0.957	2.1	0.810	25.3	0.797	11.2	0.734	30.8
SLIPPER	0.959	19.5	0.952	3.9	0.802	14	0.802	14.7	0.674	18.8
SLIPPER [14]	0.958	▨	▨	▨	0.752	▨	0.806	▨	0.745	▨
LPBoost stumps [17]	0.966	▨	▨	▨	0.795	70.8	▨	▨	0.870	85.7
LPBoost C4.5 [17]	0.959	▨	0.959	▨	0.791	▨	▨	▨	0.817	▨

Table 1: Average accuracy and $\|\lambda\|_0$ for 20 replications of 10 folds each. The bottom three rows are as reported for SLIPPER [14] and LP-Boost [17]. Gray cells indicate that the corresponding data are unavailable from the corresponding publication.

classifiers, and to SLIPPER. These data were obtained by 20 replications of 10-folding, with all three algorithms using the same training and testing sets. The bottom portion of the table shows a practical comparison of our results with the previously published results of LP-Boost with C4.5 and stump decision trees [17], and the previously published performance of the SLIPPER algorithm [14]. For LP-Boost, Demiriz *et al.* fine-tuned the parameter ν for the different datasets, so that we did not expect to match all of their classification results here. The SLIPPER algorithm uses AdABoost with a heuristic (greedy) monomial base learner. In order to prevent overfitting, the SLIPPER algorithm uses cross-validation to simplify (prune) some of the monomials that are initially generated by the greedy algorithm. We ran the SLIPPER algorithm using the publicly available version with all parameters set to their default values. The apparent discrepancy in the results of the SLIPPER algorithm in our runs may be due to the binarization of the datasets in our experiments. Finally, we expect to be able to improve the performance of L_0-RBoost by using cross validation to optimize the parameter ρ or the penalty parameters c_u.

The LP-Boost results reported in the top portion of Table 1 use $\nu = 0.50$ for $K = 5$, and $\nu = 0.56$ for $K = 1$. These LP-Boost parameters were chosen so that the resulting classifiers were quite sparse, while achieving good classification performance on the SONAR and CLHEART datasets. The results of Table 1 indicate that L_0-RBoost finds classifiers that are approximately as sparse as the classifiers found by LP-Boost, but usually achieve superior classification performance. Both algorithms seem to outperform SLIPPER. Some of the possible deficiencies of L_0-RBoost in fact occur with the less restricted class of monomials $K = 5$, suggesting that overfitting may be occurring in some cases. The possibility of overfitting may motivate us to evaluate other types of penalty functions in place of (28), along with different values of the penalty constants, in future work. Particularly, complexity measures that can be computed based on the input data, such as Rademacher complexity [5], could be the subject of further investigation.

References

[1] Sanjeev Arora, László Babai, Jaques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and Systems Sciences*, 54:317–331, 1997.

[2] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007.

[3] Egon Balas and Shu Ming NG. On the set covering polytope: I. all the facets with coefficients in $\{0, 1, 2\}$. *Mathematical Programming*, 43(1-3):57–69, 1989.

[4] Peter L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):526–536, 1998.

[5] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[6] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.

[7] Shai Ben-David, Nadav Eiron, and Philip M. Long. On the difficulty of maximizing agreements. *Journal of Computer and Systems Sciences*, 66(3):496–514, 2003.

[8] Kristin P. Bennett and Erin J. Bredensteiner. A parametric optimization method for machine learning. *INFORMS Journal of Computing*, 9(3):311–318, 1997.

[9] Kristin P. Bennett and Erin J. Bredensteiner. Duality and geometry in SVM classifers. *Proceedings of the 17th International Conference on Machine Learning*, pages 57–64, 2000.

[10] Kristin P. Bennett and Olvi L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[11] Avrim Blum and John Langford. PAC-MDL bounds. In *COLT*, pages 344–357, 2003.

[12] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, and Alexander Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.

[13] Joel Brenner. The Hadamard maximum determinant problem. *The American Mathematical Monthly*, 79(6):626–630, 1972.

[14] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *In Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 335–342, 1999.

[15] Gérard Cornuéjols and Antonio Sassano. On the 0, 1 facets of the set covering polytope. *Mathematical Programming: Series A and B*, 43(1):44–55, 1989.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[17] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[18] David P. Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *Journal of Computer and Systems Sciences*, 52(3):453–470, 1996.

[19] Jonathan Eckstein and Noam Goldberg. An improved branch-and-bound method for maximum monomial agreement. In *OPT 2008 Optimization for Machine Learning, NIPS 2008 Workshop*, 2008.

[20] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.

[21] Jerome H. Friedman. Fast sparse regression and classification. Technical report, Stanford University, 2008.

[22] Paul C. Gilmore and Ralph E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[23] Noam Goldberg and Chung-chieh Shan. Boosting optimal logical patterns. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, 2007.

[24] Thore Graepel, Ralf Herbrich, Bernhard Schölkopf, Alex Smola, Peter Bartlett, Klaus-Robert Müller, Klaus Obermayer, and Robert Williamson. Classification on proximity data with lp-machines. *International Conference of Artificial Neural Networks*, pages 304–309, 1999.

[25] David Heath. *A geometric framework for machine learning*. PhD thesis, Johns Hopkins University, 1992.

[26] Klaus-Uwe Höffgen, Hans U. Simon, and Kevin S. Van Horn. Robust trainability of single neurons. *Journal of Computer and Systems Sciences*, 50:114–125, 1995.

[27] Cong Huang, Gerald H.L. Cheang, and Andrew R. Barron. Risk of penalized least squares, greedy selection and L1 penalization for flexible function libraries. *Annals of Statistics*, Submitted 2008.

[28] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[29] Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer-Verlag, 2002.

[30] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[31] Olvi L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[32] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. *Advanced Lectures on Machine Learning*, pages 118–183, 2003.

[33] Marc E. Pfetch. Branch-and-cut for the maximum feasible subsystem problem. *SIAM Journal on Optimization*, 19:21–38, 2008.

[34] Gunnar Rätsch, T. Onoda, and K . R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320, 2001.

[35] Gunnar Rätsch, Bernhard Schölkopf, Alex J. Smola, Sebastian Mika, Takashi Onoda, and Klaus-Robert Müller. Robust ensemble learning. In Alex J. Smola, Peter J. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 2000.

[36] Robert E. Schapire. *The boosting approach to machine learning: an overview.* Springer Verlag, 2003.

[37] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26:1651–1686, 1998.

[38] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* The MIT Press, 2002.

[39] John Shawe-taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44:1926–1940, 1998.

[40] Vladimir N. Vapnik. *Statistical learning theory.* John Wiley and Sons, 1998.

[41] Vijay V. Vazirani. *Approximation algorithms.* Springer Verlag, 2003.

[42] Ulrike von Luxburg, Olivier Bousquet, and Bernhard Schölkopf. A compression approach to support vector model selection. *Journal of Machine Learning Research*, 5:293–323, 2004.

[43] Tong Zhang. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In *Neural Information Processing Systems*, 2008.