Boosting Classifiers with Tightened L_0-Relaxation Penalties

Noam Goldberg NOAMGOLD@TX.TECHNION.AC.IL
Faculty of IE&M, Technion - Israel Institute of Technology, Haifa 3200, Israel

Jonathan Eckstein JECKSTEI@RCI.RUTGERS.EDU
MSIS Department and RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway NJ 08854

Abstract

We propose a novel boosting algorithm which improves on current algorithms for weighted voting classification by striking a better balance between classification accuracy and the sparsity of the weight vector. In order to justify our optimization formulations, we first consider a novel integer linear program as a model for sparse classifier selection, generalizing the minimum disagreement halfspace 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-norm penalty. We note that common "soft margin" linear programming formulations for robust classification are equivalent to the continuous relaxation of our formulation. Since the initial continuous relaxation is weak, we suggest a tighter relaxation, using novel cutting planes, to better approximate the integer solution. To solve this relaxation, we propose a new boosting algorithm based on linear programming with dynamic generation of variables and constraints. 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 observations, 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 $y \in \{-1,1\}^M$. We have a potentially large set of base classifiers $h_u : \mathbb{R}^N \to \{-1,0,1\}$ indexed by the set $\mathcal{U} = \{1,\dots,U\}$, and would like to train a weighted voting classifier $g(x) = \sum_{u\in\mathcal{U}} \lambda_u h_u(x)$; we classify a new observation $x \in \mathbb{R}^N$ as either positive or negative based on $\operatorname{sgn}(g(x))$.

To determine λ, the empirical risk minimization strategy minimizes the sum over the observations of the 0/1 loss

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

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

$$\ell(A_i, y_i, g(A_i)) = \max\{1 - y_i g(A_i), 0\}, \tag{2}$$

and adding a model complexity penalty: a simple complexity penalty is the number of features used by the the classifier, that is, the number of $u \in \mathcal{U}$ for which $\lambda_u \neq 0$. This quantity is denoted $\|\lambda\|_0$ and called the L_0-norm (although it is not a true norm).

In order to avoid hard combinatorial optimization problems, methods such as "soft margin" LPs (Graepel et al., 1999; Rätsch et al., 2001; Demiriz et al., 2002) and SVMs (Cortes & Vapnik, 1995) use the L_1 or L_2 norms of λ, respectively, instead of L_0. This approach has the computational advantage of yielding a convex optimization problem, provided one is using an appropriate loss function such as (2).

Demiriz et al. (2002) solve the following linear programming (LP) problem, based on previous formulations of Graepel et al. (1999) and Rätsch et al. (2001):

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

$$\max_{\rho,\xi,\lambda\geq 0}\left\{\rho - D\sum_{i=1}^{M}\xi_i \;\middle|\; \begin{array}{l}\mathrm{diag}(y)H\lambda+\xi\geq\rho\mathbb{1}\\ \sum_{u=1}^{U}\lambda_u=1\end{array}\right\}, \tag{3}$$

where $\mathrm{diag}(y)$ is the diagonal matrix with entries $y_1,\ldots,y_M$. 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 $y_iH_i\lambda$, 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 $\xi_i=\max\{\rho-y_iH_i\lambda,0\}$. In (3), the objective is to maximize the margin of separation, minus a misclassification penalty; however, this formulation is known to be equivalent, for some appropriate choice of the constant D', to

$$\min_{\lambda,\xi\geq 0}\left\{\sum_{u=1}^{U}\lambda_u + D'\sum_{i=1}^{M}\xi_i \;\middle|\; \mathrm{diag}(y)H\lambda+\xi\geq\mathbb{1}\right\}; \tag{4}$$

see for example Bennett & Bredensteiner (2000). Here, the margin is fixed to 1, and one minimizes the L_1-norm of λ, plus a misclassification penalty.

The matrix H, in which each column corresponds to the vector of labels assigned by a base classifier $u\in\mathcal{U}$, usually has too many columns to be written in full as a part of the LP formulation. Instead, Demiriz et al. (2002) propose, in their LP-Boost algorithm, a column generation procedure that creates columns of H as they are needed; to be exact, they generate cuts for the dual formulation, which is essentially equivalent.

Here, we propose a related approach for an enhanced formulation of the learning problem. Instead of (4), which minimizes the sum of (2) over the observations, plus $D'\left\|\lambda\right\|_1$, we consider the combinatorial problem which minimizes the sum of the loss (1) over the observations, plus a penalty proportional to the density of the vector λ. Specifically, for any scalar $C\geq 0$, we consider the problem

$$\min_{\lambda\geq 0}\left\{\sum_{i=1}^{M}\mathbf{I}(y_iH_i\lambda\leq 0) + C\left\|\lambda\right\|_0\right\}, \tag{5}$$

Even special cases of (5) are known to be $\mathcal{NP}$-hard (see Section 2). First, we formulate the problem as a mixed integer program (MIP) and point out that the continuous relaxation of this MIP is in fact equivalent to (4). Rather than attempting to solve our MIP formulation, which is a potentially difficult combinatorial problem, or solving its weak continuous relaxation, which would be equivalent to (4) and thus to (3) and prior work, we propose a computationally tractable "middle ground": we solve a version of the continuous relaxation augmented by a large class of inequalities that are satisfied by all integer-feasible solutions of the MIP; see Section 3. We propose an algorithm which solves our relaxation formulation by dynamically generating not only variables (as in LP-Boost), but also constraints, and investigate the properties of the classifiers it constructs. Previous related work such as Weston et al. (2003) proposes methods based on finding a local minimum of a non-convex approximation of (5). Since our method's approximation of (5) is a true relaxation which may be solved to optimality, our approach, unlike this earlier work, yields a lower bound for (5), and its output is not dependent on which of potentially many local minima might be encountered.

Before exploring the details of our proposed approach, we briefly discuss some learning-theoretic motivations for starting from a formulation like (5), as well as some analysis which can suggest how to choose the penalty parameter C; these issues are explored in greater detail in Goldberg & Eckstein (2010). First, it has been known dating back to Freund & Schapire (1997) that there are bounds on the classifier prediction risk which can be expressed in terms of $\left\|\lambda\right\|_0$. Another approach to prediction risk and model selection uses the minimum prediction length (MDL) principle, which views learning as a problem of optimally compressing the data (Grünwald, 2007). The promise of using compression-based risk bounds is suggested, for example, in Von Luxburg et al. (2004) in the case of SVMs. In particular, we consider a two-part-code MDL approach which views the selection of the classifier function g as a problem of minimizing the length of a code used to transmit the vector of labels; the first part of the code describes the classifier g, and the second part encodes which observations are misclassified by g. In particular, Blum & Langford (2003) provide a classification risk bound proportional to the code length

$$\bar{L}[y,g] = L[g] + L[y|g],$$

where $L[g]$ denotes the number of bits required to encode g, and $L[y|g]$ is the number of bits needed to encode those labels y that are incorrectly predicted by g. One can derive an upper bound on $L[g]$ that is proportional to $\left\|\lambda\right\|_0$, and an upper bound on $L[y|g]$ that is proportional to $\sum_{i=1}^{M}\mathbf{I}(y_iH_i\lambda\leq 0)$. Therefore, with an appropriate choice of C, problem (5) minimizes a bound on $\bar{L}[y,g]$, which in turn corresponds to a bound on the prediction risk. In some applications, the bound on $L[g]$ can be tightened by minimizing an objective function that applies potentially different penalties c_u to different features u. We will explore such a generalization of the penalty term $C\left\|\lambda\right\|_0$ below.

2. Sparse Weighted Voting Classifier Selection (SWVC)

Finding $\lambda \in \mathbb{R}^U$ that minimizes $\sum_{i=1}^M \mathbf{I}(y_i H_i \lambda \leq 0)$ is known as the *minimum disagreement halfspace* (MDH) problem, and is $\mathcal{NP}$-hard (Höffgen et al., 1995). We call problem (5), which generalizes MDH by including an L_0 penalty and also requires $\lambda \geq 0$, the *sparse weighted voting classifier* (SWVC) problem. Requiring $\lambda \geq 0$ is without loss of generality, since at the cost of $\mathcal{U}$ being at most twice as large, we may include in $\mathcal{U}$ the negative "mirror image" $h_{u^-} = -h_u$ of any base classifier h_u. Considering the special case $C = 0$, existing MDH complexity and inapproximability results clearly also apply to SWVC. These computational complexity results also extend to other values of C (Goldberg & Eckstein, 2010); for example, the problem remains $\mathcal{NP}$-hard for $C \in O(M^{1-\epsilon})$ for all $\epsilon > 0$. We now formulate SWVC 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:

$$\min_{\xi,\mu,\lambda} \left\{ \sum_{i=1}^{M} \xi_i + C \sum_{u=1}^{U} \mu_u \;\middle|\; \begin{array}{l} (\xi,\mu,\lambda) \in Q_{H,y} \\ \xi \in \{0,1\}^M \\ \mu \in \{0,1\}^U \end{array} \right\}, \quad (6)$$

where $Q_{H,y}$ is the *soft margin classification polytope* defined as:

$$Q_{H,y} = \left\{ (\xi,\mu,\lambda) \;\middle|\; \begin{array}{l} \mathrm{diag}(y) H \lambda + (ME+1)\xi \geq \mathbb{1} \\ \lambda \leq E\mu \\ \xi \in \mathbb{R}^M, \quad \xi \geq 0 \\ \mu, \lambda \in \mathbb{R}^U, \quad \mu, \lambda \geq 0 \end{array} \right\},$$

and E is a suitably large constant. We can show that formulation (6) correctly models SWVC if $E \geq M^{M/2}$ (Goldberg & Eckstein, 2010).

We next consider the continuous relaxation of (6),

$$\min_{\xi,\mu,\lambda} \left\{ \textstyle\sum_{i=1}^M \xi_i + C \sum_{u=1}^U \mu_u \;\mid\; (\xi,\mu,\lambda) \in Q_{H,y} \right\}. \quad (7)$$

Theorem 2.1. *(ξ, λ) is an optimal solution of (4) if and only if $(\xi/(ME+1), \lambda/E, \lambda)$ is an optimal solution of (7) with penalty $C = 1/(D'(M + 1/E))$.*

Our proof of this result (Goldberg & Eckstein, 2010) is based on a one-to-one mapping, which scales objective values by a positive constant, between solutions that are feasible for (4) and solutions that are feasible for (7). An immediate consequence of Theorem 2.1 is that the SWVC relaxation (7) is equivalent to the soft margin formulation (4) for appropriate choices of the penalties C and D'. Since formulation (4) is known to be equivalent to the soft margin maximization formulation (3), it also follows that formulation (3) is equivalent to the SWVC relaxation (7).

3. Relaxing the Hard Problem and Strengthening the Relaxation

The standard continuous relaxation (7) of the MIP (6) can be very weak. Although we omit the details here, one can construct a family of examples in which the optimal integral and continuous relaxation objective values of (6) differ by a multiplicative factor of ME. In Goldberg & Eckstein (2010), we prove that one should have $E \in \Omega(2^M)$ in order to maintain the general correctness of formulation (6), implying that there are cases in which the ratio between the optimal value of (6) and the value of its continuous relaxation (7) is $\Omega(M2^M)$.

We propose to solve a natural generalization of the SWVC relaxation which applies a potentially different penalty c_u to each feature $u \in \mathcal{U}$, replacing the objective function of (7) with (8a) below. Also, for the purpose of comparison with formulation (3), we introduce the normalization constraint (8c) below, and treat the classifier margin ρ as a parameter, rather than fixing it at 1 as in (7):

$$\begin{array}{lll}
\min\limits_{\lambda,\mu,\xi} & \displaystyle\sum_{i=1}^{M} \xi_i + \sum_{u=1}^{U} c_u \mu_u & \text{(8a)} \\
\text{s.t.:} & \displaystyle\sum_{u \in \mathcal{U}} y_i H_{iu} \lambda_u + (1+\rho)\xi_i \geq \rho \quad i = 1, ..., M & \text{(8b)} \\
& \displaystyle\sum_{u \in \mathcal{U}} \lambda_u = 1 & \text{(8c)} \\
& \mu_u - \lambda_u \geq 0 \qquad u \in \mathcal{U} & \text{(8d)} \\
& \lambda_u, \mu_u, \xi_i \geq 0 \qquad \begin{array}{l} u \in \mathcal{U} \\ i = 1, ..., M. \end{array} & \text{(8e)}
\end{array}$$

This approach also eliminates the large constant E from the formulation, thus avoiding practical numerical difficulties. We now consider adding valid inequalities to (8a)-(8e) in order to strengthen its relaxation. We say that a base classifier h *distinguishes* between observations i and i' if it classifies i correctly and differently from i', that is, $h_u(A_i) = y_i \neq h_u(A_{i'})$. Let $S_{i,i'} = \{ u \in \mathcal{U} \mid h_t(A_i) = y_i \neq h_u(A_{i'}) \}$ denote the set of base classifiers that correctly classify observation i and distinguish it from i'. Let $\Phi = \{(i,i') \mid y_i \neq y_{i'}\}$ and consider the inequalities

$$\xi_i + \xi_{i'} + \sum_{u \in S_{i,i'}} \mu_u \geq 1 \qquad (i,i') \in \Phi. \quad \text{(8f)}$$

Intuitively, these inequalities assert that for each $(i,i') \in \Phi$, we must either misclassify at least one of

the observations i and i', or we need to use at least one of the distinguishing features in $S_{i,i'}$.

Theorem 3.1. *The inequalities* (8f) *hold for all integer-feasible solutions of* (8a)-(8e).

For a proof, see Goldberg (2010). Theorem 3.1 implies that appending constraints of the form (8f) to the formulation (8a)-(8e) or (7) does not eliminate any integer solutions; however, it may well eliminate solutions for which ξ or μ are not integral.

4. The L_0-RBoost Boosting Algorithm

We now describe a boosting algorithm, which we call L_0-RBoost, that effectively solves (8), including (8f). The overall approach is similar to Demiriz et al. (2002), in that we use a column generation method to iteratively generate elements of $\mathcal{U}$ as they are needed. At each iteration, the set of features is restricted to some subset $\Gamma \subseteq \mathcal{U}$ that have been generated so far, with all variables corresponding to features $u \notin \Gamma$ implicitly set to 0. Due to the potentially large number of constraints (8f), our algorithm, unlike (Demiriz et al., 2002), also dynamically generates not only variables, but also constraints. In particular, at each iteration, it solves the following LP problem:

LP($\Gamma, \mathcal{A}$): the problem (8), but fixing $\lambda_u = \mu_u = 0$ for all $u \in \mathcal{U}\backslash\Gamma$, and including constraints of the form (8f) only for (i, i') in some subset $\mathcal{A} \subseteq \Phi$.

When Γ and $\mathcal{A}$ are relatively small sets, LP($\Gamma, \mathcal{A}$) is equivalent to an LP of much smaller dimension than (8), as variables fixed to 0 may be dropped from the formulation.

Let w_i, α, and $v_{ii'}$ denote the LP dual variables (or Lagrange multipliers) of constraints (8b), (8c), and (8f), respectively. Each iteration of L_0-RBoost proceeds as follows: first, we solve LP($\Gamma, \mathcal{A}$), and next we attempt to use the resulting dual variable information to identify a feature $u^* \in \mathcal{U}$ whose addition to the set of features Γ would improve the optimal objective function value (8a). This step, which invokes the base or "weak" learning algorithm, coincides with the notion of a "pricing" step in LP column generation. The LP-Boost algorithm (Demiriz et al., 2002) operates in a similar manner, but solves (the dual of) the LP formulation (3). In LP-Boost, one would ideally like to execute the pricing/base learning step by solving the problem $\max_{u\in\mathcal{U}} \sum_{i=1}^{M} y_i H_{iu} w_i$, which is known as the *maximum agreement* problem (Kearns et al., 1994). In L_0-RBoost, the pricing problem is more general than maximum agreement, due to the additional constraints (8f) and their corresponding dual variables $v_{ii'}$: we would like to minimize, over all $u \in \mathcal{U}$, the quantity

$$\bar{c}(u) = c_u - \sum_{\substack{(i,i')\in\mathcal{A}:\\ S_{ii'}\ni u}} v_{ii'} - \sum_{i=1}^{M} y_i H_{iu} w_i - \alpha. \qquad (9)$$

If $\bar{c}(u) \geq 0$ for all $u \in \mathcal{U}$, then any optimal solution of the problem LP($\Gamma, \mathcal{A}$) is also optimal for the larger problem LP($\mathcal{U}, \mathcal{A}$) including all possible features. For brevity, we omit the formal proof of this assertion, which involves merging two constraints in the dual of LP($\mathcal{U}, \mathcal{A}$), while preserving all optimal solutions. On the other hand, if solving (9) identifies some $u^* \in \mathcal{U}\backslash\Gamma$ with $\bar{c}(u^*) < 0$, we expand Γ to include this new feature.

Next, the L_0-RBoost iteration performs an operation with no direct analog in LP-Boost: it attempts to expand the set of pairs $\mathcal{A}$ for which constraints of the form (8f) are included in the formulation. The algorithm augments $\mathcal{A}$ with pairs of observations (i, i') that are distinguished by the new feature u^*, and in some cases, when termination is imminent, by all $(i, i') \in \Phi$ for which (8f) is violated. It then proceeds to the next iteration, re-solving LP($\Gamma, \mathcal{A}$) with the expanded set Γ and $\mathcal{A}$. Algorithm 1 gives a complete description of L_0-RBoost.

Algorithm 1 L_0-RBoost

Input: $M \times N$ matrix A and labels $y \in \{-1, 1\}^M$
Output: (ξ, μ, λ)
$\Gamma \leftarrow \{1, 2\}$, where $h_1(A_i) = 1$ and $h_2(A_i) = -1$ for all $i \in \{1, \ldots, M\}$
$\mathcal{A} \leftarrow \emptyset$
repeat
  Solve LP($\Gamma, \mathcal{A}$), obtaining the solution (ξ, μ, λ) and dual variables w, α, and $v_{ii'}$, $(i, i') \in \mathcal{A}$
  Solve the base learning problem: with $\bar{c}(u)$ defined as in (9), find $u^* \in \text{Arg min}_{u\in\mathcal{U}}\{\bar{c}(u)\}$
  $\Gamma \leftarrow \Gamma \cup \{u^*\}$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{(i, i') \,|\, y_{i'} \neq y_i = h_{u^*}(A_i) \neq h_{u^*}(A_{i'})\}$
  $V \leftarrow \left\{(i, i') \;\middle|\; \xi_i + \xi_{i'} + \sum_{u\in S_{i,i'}\cap\Gamma} \mu_u < 1\right\}$
  if $\bar{c}(u^*) \geq 0$ and $V \neq \emptyset$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \cup V$
  end if
until $\bar{c}(u^*) \geq 0$ and $V = \emptyset$

Note that unlike standard column generation and LP-Boost, each iteration involves the generation of two "coupled" variables λ_{u^*} and μ_{u^*}. We initialize Γ to contain two simple elements, corresponding to the constant base classifiers that classify all observations as

respectively either positive or negative. Using LP duality theory and the construction of the set V in step 1 of the algorithm, it can be shown that when L_0-RBoost terminates, (ξ, λ, μ) is optimal for (8). Note that it would also be possible to add multiple features at each iteration, and there are techniques for terminating the algorithm early while still providing guarantees on the quality of the solution; see Lübbecke & Desrosiers (2005) and Goldberg (2010). We did not find either technique necessary in the computational experiments presented in Section 6 below.

The cuts added in step 9 are designed to increase the value of the newly generated variable μ_{u^*}, which could otherwise be as small as λ_{u^*}. In practice, it may be possible to accelerate the algorithm by adding only a carefully chosen subset of the cuts specified in step 9; see the next section for an example. Step 12 augments $\mathcal{A}$ by all currently violated cuts, ensuring that all constraints (8f) are satisfied at termination. In principle, we could include such cuts at every iteration, but we only do so if $\bar{c}(u^*) \geq 0$ and termination seems imminent. The reason is that the number of misclassified observation, which is strongly related to the number of violated cuts, tends to drop as the algorithm progresses; delaying step 12 thus allows fewer cuts to be generated in total. Finally, note that the number $|\Phi|$ of cuts (8f) is at most quadratic in the number of observations. Thus, for data sets with very few observations, it may be possible to start the algorithm with all possible cuts, initializing $\mathcal{A} = \Phi$.

5. Base Learning with Simple Rules

We now consider applying our boosting algorithm to construct ensembles of simple rules (Cohen & Singer, 1999; Friedman & Popescu, 2008). For this example, we assume that our given data matrix A is binary; note that any data matrix $A \in \mathbb{R}^{M \times N'}$ can be "binarized" using a number of attributes that is at most polynomially larger than N' (Boros et al., 1997; Goldberg & Shan, 2007). In this binary setting, we choose our base classifiers to be signed *Boolean monomials*, that is, functions $h_{u^+}, h_{u^-} : \{0,1\}^N \to \{-1, 0, 1\}$ given by

$$h_{u^+}(x) = \prod_{j \in J} x_j \prod_{c \in C} (1 - x_c) \quad \text{and} \quad h_{u^-} = -h_{u^+},$$

with $J, C \subseteq \{1, \ldots, N\}$ and $J \cap C = \emptyset$. The *degree* of such a monomial is simply $|J| + |C|$.

When the set of possible features $\mathcal{U}$ corresponds to all possible monomial classifiers over the N variables, minimizing the function (9) is a generalization of the $\mathcal{NP}$-hard *maximum monomial agreement* problem (Kearns et al., 1994). However, in most learning applications, it is reasonable to bound the degree of the monomials considered by a constant K; the monomial that maximally agrees with the data, in the sense of minimizing (9), can then be found in polynomial time. The base learning problem minimizing (9) can be solved by trivial enumeration for simple classes of base classifiers, such as monomials of some given small degree or trees of small depth (*e.g.*, "decision stumps"). For monomials of a larger degree, we experiment in Section 6 with an extension of the maximum monomial agreement algorithm of Eckstein & Goldberg (2008); for more details, see Goldberg (2010).

The MDL/compression motivation discussed in Section 1, together with the structural risk minimization (SRM) approach to learning (Vapnik, 1998; Shawe-Taylor et al., 1998), suggests an encoding scheme in which the complexity penalty assigned to a monomial feature varies with its degree. In particular, we set the cost c_u of using feature u to

$$c_u = \frac{k + \log\binom{N}{k} + \log K}{\log M} + \kappa, \tag{10}$$

where k is the degree of the monomial corresponding to u, and κ is a small positive constant. A detailed account of this choice may be found in Goldberg & Eckstein (2010), but in brief the motivation is as follows: we first encode the degree k of the monomial corresponding to u, requiring approximately $\log K$ bits since the maximum allowed degree is K. Then, we encode which of the $2^k\binom{N}{k}$ possible monomials of degree k is meant, which requires an additional $\log\left(2^k\binom{N}{k}\right)$ bits. We then divide this quantity by $\log M$, which is an upper bound on the number of bits needed to encode a misclassified observation; this normalization is required since the objective coefficients of the misclassification variables ξ_i in (8a) are all 1.

In our experiments applying Algorithm 1 to Boolean monomial base classifiers, we found it beneficial to add only a subset of the possible cuts in step 9. In particular, we found a small Hamming distance between A_i and $A_{i'}$ to be a good indicator of the potential of cut (i, i') to tighten the relaxation. After adjoining the base classifier u^* to the formulation, we scan all pairs (i, i') for which $y_i = h_{u^*}(A_i) \neq h_{u^*}(A_{i'})$, and use only the cuts corresponding to pairs whose Hamming distance is within a fixed factor of the minimum encountered. This technique reduces the amount of time spent adding new rows to the LP constraint matrix (which can be time-consuming, since LP solvers typically store the constraint matrix in a column-oriented sparse form), and restrains the size of the LP. Steps 10-13 of the algorithm still ensure that there are no remaining violated constraints upon termination.

6. Experimental Work and Discussion

Using Boolean monomial base classifiers of maximum degree $K = 1$ or $K = 5$, we compared the classification performance of L_0-RBoost (Algorithm 1) to LP-Boost (Demiriz et al., 2002). We experimented with five UCI datasets (Asuncion & Newman, 2007): BCW, VOTE, CLVHEART, HUHEART, and SONAR. Observations with missing attribute values were deleted, except for the congressional voting dataset (VOTE), where a missing value was considered as a distinct attribute value. The data were binarized using the technique of Boros et al. (1997); the resulting number of attributes ranged between 8 and 17, and varied by cross-validation (CV) run as well as by dataset.

The optimization formulation used by LP-Boost is (3) with $D = 1/\nu M$. In this work, the parameter ν corresponds to an upper bound on the fraction of margin errors $|\{i \mid \xi_i > 0\}|/M$ (Graepel et al., 1999); however, no method for choosing ν is suggested other than intensive cross-validation, and making use of the classification results of other algorithms. The LP formulation (3) is known to find sparse classifiers, but is also sensitive to the choice of the tunable penalty parameter D. By choosing a small enough penalty D (*i.e.*, a large enough ν) in (3), it is possible to find a sparse "degenerate" classifier with large ρ by assigning a large proportion of observations to be outliers.

In our experiments with L_0-RBoost, we set the parameters c_u of formulation (8) using (10) with $\kappa = 1.5$. We then investigated the dependence of L_0-RBoost and LP-Boost on the tunable parameters ρ and ν, respectively, examining both classifier performance and sparsity. Figure 1 displays the the training accuracy, testing accuracy, and $\|\lambda\|_0$ attained by L_0-RBoost and LP-Boost on the SONAR dataset, in relation to the algorithms' tunable parameters, for monomials of degree $K = 1$ or up to $K = 5$. Each plot point corresponds to 10 replications of a 10-fold CV experiment. For both algorithms, the figure shows that the classifiers selected tend to overfit the training data for small values of the parameters. This overfitting is by far the most pronounced for LP-Boost with $K = 5$. The experiments show that, over the entire range of parameter values, L_0-RBoost generalizes well compared with LP-Boost. We also find that the performance of L_0-RBoost is robust with respect to a wide range of choices of the parameter ρ. In LP-Boost, the margin ρ rises monotonically as one increases ν; so, in Figure 1, we show ρ rather than ν on the horizontal axis to facilitate comparison with L_0-RBoost. We found the performance of LP-Boost to be highly sensitive to the choice of the parameter, confirming the observations of Demiriz et al. (2002). When the input data are not linearly separable by the set of base classifiers $\mathcal{U}$, as is typically the case when $K = 1$, and ν is small, then a non-separating hyperplane with zero margin becomes optimal. Further, the value of ν that may be considered too small varied significantly between datasets.

Figure 2 plots accuracy on the test set versus $\|\lambda\|_0$, again for the SONAR dataset, summarizing the classification versus sparsity performance of the experiments depicted in Figure 1. As in Figure 1, each point corresponds to an average over 10 replications of a 10-fold CV 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 are fixed identically in all of the experiments. However, it is apparent that the classifiers computed by L_0-RBoost tend to strike a better trade-off between accuracy and sparsity than the classifiers computed by LP-Boost.

Table 1 compares the average classification accuracy and sparsity obtained by the classifiers computed by L_0-RBoost, LP-Boost, and SLIPPER (Cohen & Singer, 1999). SLIPPER combines AdaBoost with a heuristic base learner; in our case of binary data, the rules produced by this heuristic are simply monomials of arbitrary degree. The table entries in the top five rows are our own computational results for 20 replications of 10-fold CV experiments. In these experiments we compare L_0-RBoost, with $K = 1$ or $K = 5$, to LP-Boost using the same base classifiers, and to SLIPPER with monomials of arbitrary degree. The L_0-RBoost results in the table use the fixed parameter value $\rho = 20/M$, while the LP-Boost results in this top portion of the table use $\nu = 0.50$ for $K = 5$, and $\nu = 0.56$ for $K = 1$. These values of ν were chosen because they produced relatively sparse classifiers which seemed to perform the best on the SONAR and CLVHEART datasets. The bottom portion of the table shows a practical comparison of our results with the previously published results for LP-Boost with C4.5 and stump decision trees (Demiriz et al., 2002), and the previously published performance of the SLIPPER algorithm (Cohen & Singer, 1999), which were also based on a 10-fold CV setup. The LP-Boost results of Demiriz et al. (2002) use values of ν individually tuned for each dataset, so we did not expect to match all of their classification results. We ran the SLIPPER algorithm using the publicly available implementation, with all parameters set to their default values. The apparent discrepancy between our observation of SLIPPER's performance and the previously published results may be due to the binarization of the datasets in our experiments.

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. Finally, although we demonstrate this classification performance based on parameters that are justified analytically, without fine-tuning, we can only expect to improve the performance of L_0-RBoost by using cross-validation to select the margin parameter ρ or the penalty parameters c_u.

Acknowledgments Based upon work funded in part by the U.S. DHS under Grant Award #2008-DN-077-ARI001-02. We thank Rob Schapire for helpful discussions and comments. The first author would also like to thank Kristin Bennett for comments at an early stage of this work.

References

Asuncion, A. and Newman, D.J. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Bennett, K.P. and Bredensteiner, E.J. Duality and geometry in SVM classifers. *Proc. of the 17th Int. Conf. on Machine Learning*, pp. 57–64, 2000.

Blum, A. and Langford, J. PAC-MDL bounds. In *Proc. of the 16th Annual Conf. on Computational Learning Theory*, pp. 344–357, 2003.

Boros, E., Hammer, P.L., Ibaraki, T., and Kogan, A. Logical analysis of numerical data. *Math. Programming*, 79:163–190, 1997.

Cohen, W.W. and Singer, Y. A simple, fast, and effective rule learner. In *Proc. of the 16th National Conf. on Artificial Intelligence*, pp. 335–342, 1999.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

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

Eckstein, J. and Goldberg, N. An improved branch-and-bound method for maximum monomial agreement. In *Optimization for Machine Learning, NIPS Workshop*, 2008. URL `http://opt2008.kyb.tuebingen.mpg.de/eckstein.pdf`.

Freund, Y. and Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Computer and Systems Sciences*, 55(1):119–139, 1997.

Friedman, J.H. and Popescu, B.E. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2 (3):916–954, 2008.

Goldberg, N. *Optimization for Sparse and Accurate Classifiers*. PhD thesis, Rutgers University, 2010.

Goldberg, N. and Eckstein, J. Sparse weighted voting classifier selection and its LP relaxation. Technical Report RRR 9-2010, RUTCOR, Rutgers University, 2010.

Goldberg, N. and Shan, C. Boosting optimal logical patterns. In *Proc. of SIAM Int. Conf. on Data Mining*, 2007.

Graepel, T., Herbrich, R., Schölkopf, B., Smola, A., Bartlett, P., Müller, K-R., Obermayer, K., and Williamson, R. Classification on proximity data with LP-machines. *Int. Conf. of Artificial Neural Networks*, pp. 304–309, 1999.

Grünwald, Peter D. *The minimum description length principle*. MIT Press, 2007.

Höffgen, K-U., Simon, H.U., and Horn, K.S. Van. Robust trainability of single neurons. *J. of Computer and Systems Sciences*, 50:114–125, 1995.

Kearns, M.J., Schapire, R.E., and Sellie, L.M. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

Lübbecke, M.E. and Desrosiers, J. Selected topics in column generation. *Operations Research*, 53(6): 1007–1023, 2005.

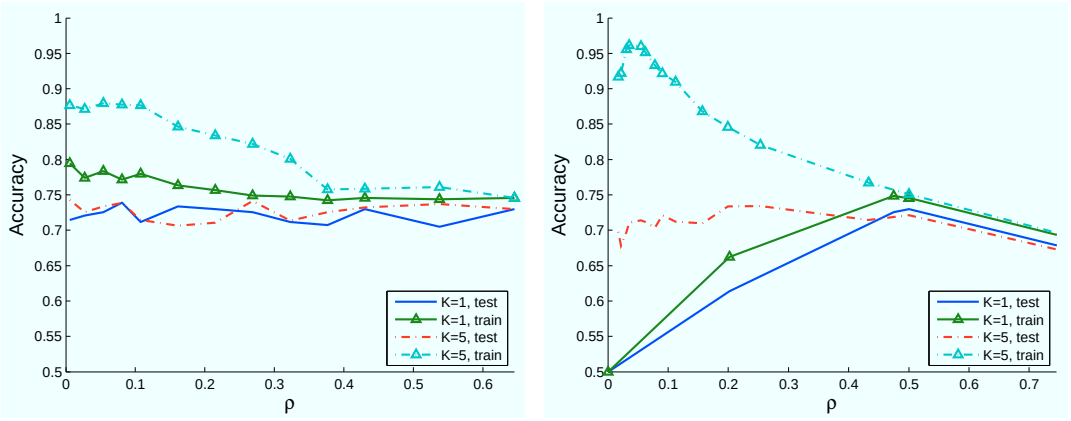
Rätsch, G., Onoda, T., and Müller, K-R. Soft margins for AdaBoost. *Machine Learning*, 42:287–320, 2001.

Shawe-Taylor, J., Bartlett, P.L., Williamson, R.C., and Anthony, M. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, 44:1926–1940, 1998.

Vapnik, V.N. *Statistical learning theory*. John Wiley and Sons, 1998.

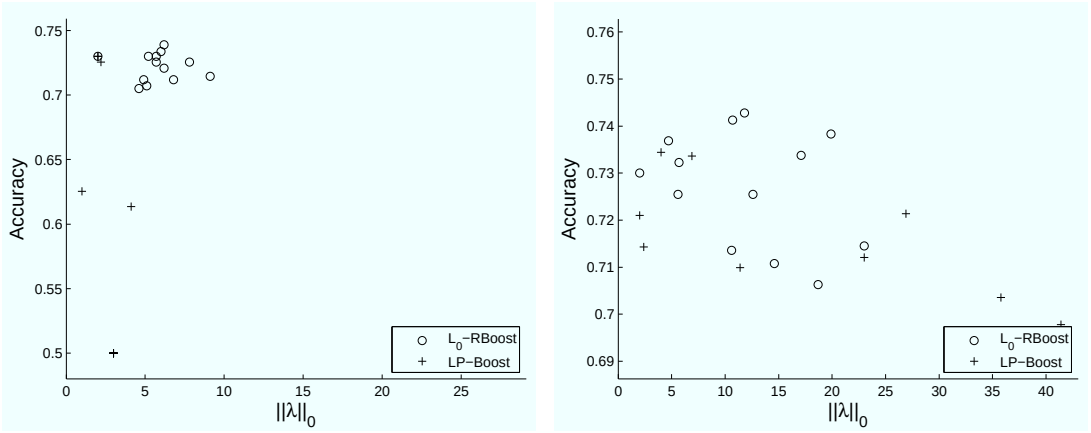
Von Luxburg, U., Bousquet, O., and Schölkopf, B. A compression approach to support vector model selection. *J. of Machine Learning Research*, 5:293–323, 2004.

Weston, J., Elisseeff, A., Schölkopf, B., and Tipping, M. Use of the zero norm with linear models and kernel methods. *J. of Machine Learning Research*, 3:1439–1461, 2003.



(a) L_0-RBoost accuracy vs. margin (b) LP-Boost accuracy vs. margin

Figure 1. The classification performance of LP-Boost versus L_0-RBoost with monomial base classifiers for the SONAR dataset. Each point corresponds to the average of a 10-replication, 10-fold CV experiment.



(a) $K = 1$ (b) $K = 5$

Figure 2. Test accuracy vs. $\|\lambda\|_0$ on the SONAR dataset: each point corresponds to an average over 10 replications of a 10-fold CV experiment with a particular input parameter

	Dataset									
Name	BCW		VOTE		CLVHEART		HUHEART		SONAR	
Samples	683		435		297		294		208	
Method	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$	Acc	$\|\lambda\|_0$
L_0-RBoost $K = 1$	0.963	9.1	0.950	6.0	0.846	10.3	0.812	10.4	0.712	7.2
L_0-RBoost $K = 5$	0.950	9.4	0.960	3.0	0.833	38.9	0.807	27.4	0.725	21.3
LP-Boost $K = 1$	0.925	3.8	0.957	2	0.774	7.6	0.803	3.8	0.735	8.6
LP-Boost $K = 5$	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	0.958				0.752		0.806		0.745	
LP-Boost stumps	0.966				0.795	70.8			0.870	85.7
LP-Boost C4.5	0.959		0.959		0.791				0.817	

Table 1. Average accuracy and $\|\lambda\|_0$ for 20 replications of 10-fold cross-validation. The bottom three rows are as reported for SLIPPER (Cohen & Singer, 1999) and LP-Boost (Demiriz et al., 2002). Gray cells indicate that the corresponding data are unavailable from the given publication.