A Column Generation Algorithm For Boosting

Kristin P. Bennett BENNEK@RPI.EDU
Dept. of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180 USA
while visiting Microsoft Research, Redmond, WA USA

Ayhan Demiriz DEMIRA@RPI.EDU
Dept. of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180 USA

John Shawe-Taylor JST@DCS.RHBNC.AC.UK
Dept. of Computer Science, Royal Holloway, University of London Egham, Surrey TW20 0EX, UK

Abstract

We examine linear program (LP) approaches to boosting and demonstrate their efficient solution using LPBoost, a column generation simplex method. We prove that minimizing the soft margin error function (equivalent to solving an LP) directly optimizes a generalization error bound. LPBoost can be used to solve any boosting LP by iteratively optimizing the dual classification costs in a restricted LP and dynamically generating weak learners to make new LP columns. Unlike gradient boosting algorithms, LPBoost converges finitely to a global solution using well defined stopping criteria. Computationally, LPBoost finds very sparse solutions as good as or better than those found by ADABoost using comparable computation.

1. Introduction

Recent papers (Schapire et al., 1998) have shown that boosting, arcing, and related ensemble methods (hereafter summarized as boosting) can be viewed as margin maximization in function space. By changing the cost function, different boosting methods such as AdaBoost can be viewed as gradient descent to minimize this cost function. Some authors have noted the possibility of choosing cost functions that can be formulated as linear programs (LP) but then dismiss the approach as intractable using standard LP algorithms (Ratsch et al., 1999; Breiman, 1999). In this paper we show that LP boosting is computationally feasible using a classic column generation simplex algorithm (Nash & Sofer, 1996). This method performs tractable boosting using any cost function expressible as an LP. We specifically examine the variations of the 1-norm soft margin cost function used for support vector machines (Ratsch et al., 2000; Bennett, 1999; Mangasarian, 2000).

In Section 2, we prove that this approach directly minimizes a bound on the generalization error. In Section 3, we discuss the soft margin LP formulation and provide an insightful explanation of how it works for boosting. In Section 4, we describe the column generation algorithm. Computational results and practical issues for implementation of the method are given in Section 5.

2. Motivation for Soft Margin Boosting

The function classes that we will be considering are of the form $\mathrm{co}(H) = \left\{\sum_{h\in H} a_h h : a_h \geq 0\right\}$, where H is a set of weak learners which we assume is closed under complementation. Initially, these will be classification functions with outputs in the set $\{-1, 1\}$, though this can be taken as $[-1, 1]$ in confidence-rated boosting. We begin, however, by looking at a general function class and quoting a bound on the generalization error in terms of the margin and covering numbers. We first introduce some notation. If $\mathcal{D}$ is a distribution on inputs and targets, $X \times \{-1, 1\}$, we define the error $\mathrm{err}_{\mathcal{D}}(\mathrm{f})$ of a function $f \in \mathcal{F}$ to be the probability $\mathcal{D}\{(\mathbf{x}, y) : \mathrm{sgn}(f(\mathbf{x})) \neq y\}$, where we assume that we obtain a classification function by thresholding at 0 if f is real-valued.

Definition 2.1. *Let $\mathcal{F}$ be a class of real-valued functions on a domain X. A γ-cover of $\mathcal{F}$ with respect to a sequence of inputs $S = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$ is a finite set of functions A such that for all $f \in \mathcal{F}$, there exists $g \in A$, such that $\max_{1\leq i\leq m} (|f(\mathbf{x}_i) - g(\mathbf{x}_i)|) < \gamma$. The size of the smallest such cover is denoted by $\mathcal{N}(\mathcal{F}, S, \gamma)$,* *while the* covering numbers *of* $\mathcal{F}$ *are the values*

$$\mathcal{N}(\mathcal{F}, m, \gamma) = \max_{S \in X^m} \mathcal{N}(\mathcal{F}, S, \gamma).$$

Throughout the remainder of this section we will assume a training set $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$. For a real-valued function $f \in \mathcal{F}$ we define the margin of an example $(\mathbf{x}, y)$ to be $yf(\mathbf{x})$, where again we implicitly assume that we are thresholding at 0. The margin of the training set S is defined to be $m_S(f) = \min_{1 \leq i \leq m} (y_i f(\mathbf{x}_i))$.

Note that this quantity is positive if the function correctly classifies all of the training examples. The following theorem is given in (Cristianini & Shawe-Taylor, 2000) but is implicit in the results of (Shawe-Taylor et al., 1998).

Theorem 2.1. *Consider thresholding a real-valued function space* $\mathcal{F}$ *and fix* $\gamma \in \mathbb{R}^+$*. For any probability distribution* $\mathcal{D}$ *on* $X \times \{-1, 1\}$*, with probability* $1 - \delta$ *over* m *random examples* S*, any hypothesis* $f \in \mathcal{F}$ *that has margin* $m_S(f) \geq \gamma$ *on* S *has error no more than*

$$\mathrm{err}_{\mathcal{D}}(\mathrm{f}) \leq \varepsilon(\mathrm{m}, \mathcal{F}, \delta, \gamma) = \frac{2}{\mathrm{m}} \left(\log \mathcal{N}(\mathcal{F}, 2\mathrm{m}, \frac{\gamma}{2}) + \log \frac{2}{\delta} \right),$$

provided $m > 2/\varepsilon$.

We now describe a construction originally proposed in (Shawe-Taylor & Cristianini, 1999) for applying this result to cases where not all the points attain the margin γ. Let X be a Hilbert space. We define the following inner product space derived from X.

Definition 2.2. *Let* $L(X)$ *be the set of real-valued functions* f *on* X *with countable support* $supp(f)$*, that is, functions in* $L(X)$ *are non-zero only for countably many points. Formally, we require*

$$L(X) = \left\{ f \in \mathbb{R}^{\mathbb{X}} : \begin{array}{l} supp(f) \text{ is countable and} \\ \sum_{\mathbf{x} \in supp(f)} f(\mathbf{x})^2 < \infty \end{array} \right\}.$$

We define the inner product of two functions $f, g \in L(X)$ *by* $\langle f \cdot g \rangle = \sum_{\mathbf{x} \in supp(f)} f(\mathbf{x}) g(\mathbf{x})$*. This implicitly defines a norm* $\| \cdot \|_2$*. We also introduce*

$$\|f\|_1 = \sum_{\mathbf{x} \in supp(f)} |f(\mathbf{x})|.$$

Note that the sum that defines the inner product is well-defined by the Cauchy-Schwarz inequality. Clearly the space is closed under addition and multiplication by scalars. Furthermore, the inner product is linear in both arguments.

We now form the product space $X \times L(X)$ with corresponding function class $\mathcal{F} \times L(X)$ acting on $X \times L(X)$ via the composition rule

$$(f, g) : (\mathbf{x}, h) \longmapsto f(\mathbf{x}) + \langle g \cdot h \rangle .$$

Now for any fixed $1 \geq \Delta > 0$ we define an embedding of X into the product space $X \times L(X)$ as follows:

$$\tau_\Delta : \mathbf{x} \longmapsto (\mathbf{x}, \Delta \delta_{\mathbf{x}}),$$

where $\delta_{\mathbf{x}} \in L(X)$ is defined by $\delta_{\mathbf{x}}(\mathbf{y}) = 1$ if $\mathbf{y} = \mathbf{x}$, and 0 otherwise.

Definition 2.3. *Consider using a class* $\mathcal{F}$ *of real-valued functions on an input space* X *for classification by thresholding at* 0*. We define the* margin slack variable *of an example* $(\mathbf{x}_i, y_i) \in X \times \{-1, 1\}$ *with respect to a function* $f \in \mathcal{F}$ *and target margin* γ *to be the quantity* $\xi\left((\mathbf{x}_i, y_i), f, \gamma\right) = \xi_i = \max\left(0, \gamma - y_i f(\mathbf{x}_i)\right)$*. Note that* $\xi_i > \gamma$ *implies incorrect classification of* $(\mathbf{x}_i, y_i)$*.*

The construction of the space $X \times L(X)$ allows us to obtain a margin separation of γ by using an auxiliary function defined in terms of the margin slack variables. For a function f and target margin γ the auxiliary function with respect to the training set S is

$$g_f = \frac{1}{\Delta} \sum_{i=1}^{m} \xi\left((\mathbf{x}_i, y_i), f, \gamma\right) y_i \delta_{\mathbf{x}_i} = \frac{1}{\Delta} \sum_{i=1}^{m} \xi_i y_i \delta_{\mathbf{x}_i}.$$

It is now a simple calculation to check the following two properties of the function $(f, g_f) \in \mathcal{F} \times L(X)$:

1. (f, g_f) has margin γ on the training set $\tau_\Delta(S)$.
2. $(f, g_f)\tau_\Delta(\mathbf{x}) = f(\mathbf{x})$ for $\mathbf{x} \notin S$.

Together these facts imply that the generalization error of f can be assessed by applying the large margin theorem to (f, g_f). This gives the following theorem:

Theorem 2.2. *Consider thresholding a real-valued function space* $\mathcal{F}$ *on the domain* X*. Fix* $\gamma \in \mathbb{R}^+$ *and choose* $\mathcal{G} \subset \mathcal{F} \times L(X)$*. For any probability distribution* $\mathcal{D}$ *on* $X \times \{-1, 1\}$*, with probability* $1 - \delta$ *over* m *random examples* S*, any hypothesis* $f \in \mathcal{F}$ *for which* $(f, g_f) \in \mathcal{G}$ *has generalization error no more than*

$$\mathrm{err}_{\mathcal{D}}(\mathrm{f}) \leq \varepsilon(\mathrm{m}, \mathcal{F}, \delta, \gamma) = \frac{2}{\mathrm{m}} \left(\log \mathcal{N}(\mathcal{G}, 2\mathrm{m}, \frac{\gamma}{2}) + \log \frac{2}{\delta} \right),$$

provided $m > 2/\varepsilon$*, and there is no discrete probability on misclassified training points.*

We are now in a position to apply these results to our function class which will be in the form described above, $\mathcal{F} = \mathrm{co}(H) = \left\{\sum_{h\in H} a_h h : a_h \geq 0\right\}$, where we have left open for the time being what the class H of weak learners might contain. The sets $\mathcal{G}$ of Theorem 2.2 will be chosen as follows:

$$\mathcal{G}_B = \left\{ \left(\sum_{h\in H} a_h h, g \right) : \sum_{h\in H} a_h + \|g\|_1 \leq B,\, a_h \geq 0 \right\}.$$

Hence, the condition that a function $f = \sum_{h\in H} a_h h$ satisfies the conditions of Theorem 2.2 for $\mathcal{G} = \mathcal{G}_B$ is simply

$$\begin{aligned} &\textstyle\sum_{h\in H} a_h + \frac{1}{\Delta}\sum_{i=1}^m \xi\left((\mathbf{x}_i, y_i), f, \gamma\right) \\ &\quad = \textstyle\sum_{h\in H} a_h + \frac{1}{\Delta}\sum_{i=1}^m \xi_i \leq B. \end{aligned} \tag{1}$$

Note that this will be the quantity that we will minimize through the boosting iterations described in later sections, where we will use the parameter C in place of $1/\Delta$ and the margin γ will be set to 1. The final piece of the puzzle that we require to apply Theorem 2.2 is a bound on the covering numbers of $\mathcal{G}_B$ in terms of the class of weak learners H, the bound B, and the margin γ. Before launching into this analysis, observe that for any input $\mathbf{x}$,

$$\max_{h\in H}\{|h(\mathbf{x})|\} = 1, \quad \text{while} \quad \max_{\mathbf{x}_i} \Delta\delta_{\mathbf{x}_i}(\mathbf{x}) \leq \Delta \leq 1.$$

2.1 Covering Numbers of Convex Hulls

In this subsection we analyze the covering numbers $\mathcal{N}(\mathcal{G}_B, m, \gamma)$ of the set

$$\mathcal{G}_B = \left\{ \left(\sum_{h\in H} a_h h, g \right) : \sum_{h\in H} a_h + \|g\|_1 \leq B,\, a_h \geq 0 \right\}$$

in terms of B, the class H, and the scale γ. Assume first that we have an η/B-cover G of the function class H with respect to the set $S = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ for some $\eta < \gamma$. If H is a class of binary-valued functions then we will take η to be zero and G will be the set of dichotomies that can be realized by the class. Now consider the set V of vectors of positive real numbers indexed by $G \cup \{1, \ldots, m\}$. Let $\mathcal{V}_B$ be the function class $\mathcal{V}_B = \{\mathbf{g} \mapsto \langle \mathbf{g}\cdot v\rangle : v \in V, \|v\|_1 \leq B, \|\mathbf{g}\|_\infty \leq 1\}$, and suppose that U be an $(\gamma - \eta)$-cover of $\mathcal{V}_B$. We claim that the set

$$A = \left\{ \left(\sum_{h\in G} v_h h, \sum_{i=1}^m v_i \delta_{\mathbf{x}_i} \right) : v \in U \right\}$$

is a γ-cover of $\mathcal{G}_B$ with respect to the set $\tau_\Delta(S)$. We prove this assertion by taking a general function $f = \left(\sum_{h\in H} a_h h, g\right) \in \mathcal{G}_B$, and finding a function in A within γ of it on all of the points $\tau_\Delta(\mathbf{x}_i)$. First, for each h with non-zero coefficient a_h, select $\hat{h} \in G$, such that $\left|h(\mathbf{x}_i) - \hat{h}(\mathbf{x}_i)\right| \leq \eta/B$, and for $h' \in G$ set $v_{h'} = \sum_{h:\hat{h}=h'} a_h$ and $v_i = g(\mathbf{x}_i)/\Delta$, $i = 1, \ldots, m$. Now we form the function $\bar{f} = \left(\sum_{h\in G} v_h h, \sum_{i=1}^m v_i \delta_{\mathbf{x}_i}\right)$, which lies in the set $\mathcal{V}_B$, since $\sum_{h\in G} a_h + \sum_{i=1}^m v_i \leq B$. Furthermore we have that

$$\begin{aligned} &\left|f(\tau_\Delta(\mathbf{x}_j)) - \bar{f}(\tau_\Delta(\mathbf{x}_j))\right| \\ &\quad = \left|\textstyle\sum_{h\in H} a_h h(\mathbf{x}_j) + g(\mathbf{x}_j) - \sum_{h\in G} v_h h(\mathbf{x}_j) - \Delta v_j\right| \\ &\quad \leq \left|\textstyle\sum_{h\in H} a_h \left(h(\mathbf{x}_j) - \hat{h}(\mathbf{x}_j)\right)\right| \\ &\quad \leq \textstyle\frac{\eta}{B}\sum_{h\in H} a_h \leq \eta \end{aligned}$$

Since U is a $\gamma - \eta$ cover of $\mathcal{V}_B$ there exists $\hat{v} \in U$ such that $\hat{f} = \left(\sum_{h\in G} \hat{v}_h h, \sum_{i=1}^m \hat{v}_i \delta_{\mathbf{x}_i}\right)$ is within $\gamma - \eta$ of $\bar{f}$ on $\tau_\Delta(\mathbf{x}_j)$, $j = 1, \ldots, m$. It follows that $\hat{f}$ is within γ of f on this same set. Hence, A forms a γ cover of the class $\mathcal{G}_B$. We bound $|A| = |U|$ using the following theorem due to (Zhang, 1999), though a slightly weaker version can also be found in (Anthony & Bartlett, 1999).

Theorem 2.3. *(Zhang, 1999) For the class $\mathcal{V}_B$ defined above we have that*

$$\begin{aligned} \log \mathcal{N}(\mathcal{V}_B, m, \gamma) \quad \leq \quad & 1 + \frac{144B^2}{\gamma^2}\left(2 + \ln(|G| + m)\right) \\ & \log\left(2\left\lceil \frac{4B}{\gamma} + 2\right\rceil m + 1\right). \end{aligned}$$

Hence we see that optimizing B directly optimizes the relevant covering number bound and hence the generalization bound given in Theorem 2.2 with $\mathcal{G} = \mathcal{G}_B$. Note that in the cases considered $|G|$ is just the growth function $B_H(m)$ of the class H of weak learners.

3. LP Approaches to Boosting

The quantity B defined in Equation (1) can be optimized directly using an LP. The LP is formulated as if all possible labelings of the training data by the weak learners were known. The LP minimizes the 1-norm soft margin cost function used in support vector machines with the added restrictions that all the weights are positive and the threshold is assumed to be zero. This LP and variants can be practically solved using a column generation approach. Weak learners are generated as needed to produce the optimal support vector machine based on the output of all the weak learners. In essence the base learner become an 'oracle' that generates the necessary columns. The dual variables of the linear program provide the classification costs needed by the learning machine. The column generation procedure searches for the best possible classification costs in dual space. Only at optimality is the actual ensemble of weak learners constructed.

3.1 LP Formulation

Let the matrix H be a m by n matrix of all the labelings of the training data using functions from $\mathcal{H}$. Specifically $H_{ij} = h_j(x_i)$ is the label (1 *or* -1) given by weak learner $h_j \in \mathcal{H}$ on the training point x_i. Each column $H_{.j}$ of the matrix H constitutes the output of weak learner h_j on the training data, while each row H_i gives the outputs of all the weak learners on the example x_i. There may be up to 2^m distinct weak learners.

The following linear program can be used to minimize the quantity in Equation (1):

$$\begin{array}{ll} \min_{a,\xi} & \sum_{i=1}^{n} a_i + C\sum_{i=1}^{m} \xi_i \\ s.t. & y_i H_i a + \xi_i \geq 1, \quad \xi_i \geq 0, \ i = 1, \ldots, m \\ & a_j \geq 0, \ i = 1, \ldots, n \end{array} \tag{2}$$

where $C > 0$ is the tradeoff parameter between misclassification error and margin maximization. The dual of LP (2) is

$$\begin{array}{ll} \max_{u} & \sum_{i=1}^{m} u_i \\ s.t. & \sum_{i=1}^{m} u_i y_i H_{ij} \leq 1, \ j = 1, \ldots, n \\ & 0 \leq u_i \leq C, \ i = 1, \ldots, m \end{array} \tag{3}$$

Alternative soft margin LP formulations exist, such as this one for the ν-LP Boosting[1]. (Ratsch et al., 1999):

$$\begin{array}{ll} \max_{a,\xi,\rho} & \rho - D\sum_{i=1}^{m} \xi_i \\ s.t. & y_i H_i a + \xi_i \geq \rho, \ i = 1, \ldots, m \\ & \sum_{i=1}^{n} a_i = 1, \quad \xi_i \geq 0, \ i = 1, \ldots, m \\ & a_j \geq 0, \ i = 1, \ldots, n \end{array} \tag{4}$$

The dual of this LP (4) is:

$$\begin{array}{ll} \min_{u} & \beta \\ s.t. & \sum_{i=1}^{m} u_i y_i H_{ij} \leq \beta, \ j = 1, \ldots, n \\ & \sum_{i=1}^{m} u_i = 1, \ 0 \leq u_i \leq D, \ i = 1, \ldots, m \end{array} \tag{5}$$

These LP formulations are exactly equivalent given the appropriate choice of the parameters C and D.

Theorem 3.1 (LP Formulation Equivalence). *If LP (4) with parameter D has a primal solution* $(\bar{a}, \bar{\rho} > 0, \bar{\xi})$ *and dual solution solution* $(\bar{u}, \bar{\beta})$*, then* $(\hat{a} = \frac{\bar{a}}{\bar{\rho}}, \hat{\xi} = \frac{\bar{\xi}}{\bar{\rho}})$ *and* $(\hat{u} = \frac{\bar{u}}{\bar{\beta}})$ *are the primal and dual solutions of LP (2) with parameter* $C = \frac{D}{\bar{\beta}}$*. Similarly, if LP 2 with parameter C has primal solution* $(\hat{a} \neq 0, \hat{\xi})$ *and dual solution solution* $(\hat{u} \neq 0)$*, then* $(\bar{\rho} = \frac{1}{\sum_{i=1}^{n} \hat{a}_i}, \bar{a} = \hat{a}\bar{\rho}, \bar{\xi} = \hat{\xi}\bar{\rho})$ *and* $(\bar{\beta} = \frac{1}{\sum_{i=1}^{m} \hat{u}_i}, \bar{u} = \hat{u}\bar{\beta})$ *are the primal and dual solutions of LP (4) with parameter* $D = C\hat{\beta}$.

[1] We remove the constraint $\rho \geq 0$ since $\rho > 0$ at optimality under the complementation assumption.

Proof. The necessary and sufficient conditions for LP optimality are primal feasibility, dual feasibility, and equality of the primal and dual objectives. For the optimal solutions of LP (4): $(\bar{a}, \bar{\rho}, \bar{\xi})$ and $(\bar{u}, \bar{\beta})$, primal feasibility of LP (4) implies primal feasibility of LP (2):

$$\begin{aligned} y_i H_i \frac{\bar{a}}{\bar{\rho}} + \frac{\bar{\xi}_i}{\bar{\rho}} \geq \frac{\bar{\rho}}{\bar{\rho}}, \ i = 1, \ldots, m \\ \frac{\bar{\xi}_i}{\bar{\rho}} \geq 0, \ i = 1, \ldots, m \\ \frac{\bar{a}_j}{\bar{\rho}} \geq 0, \ j = 1, \ldots, n. \end{aligned}$$

Dual feasibility of (5) implies dual feasibility (3):

$$\begin{aligned} \sum_{i=1}^{m} \frac{\bar{u}_i}{\bar{\beta}} y_i H_{ij} \leq \frac{\bar{\beta}}{\bar{\beta}}, \ j = 1, \ldots, n \\ 0 \leq \frac{\bar{u}_i}{\bar{\beta}} \leq \frac{D}{\bar{\beta}}, \ i = 1, \ldots, m. \end{aligned}$$

Using the equality of the objectives, $\bar{\rho} - D\sum_{i=1}^{m} \bar{\xi}_i = \bar{\beta}$, and substitution, $\sum_{i=1}^{n} \hat{a}_i + C\sum_{i=1}^{m} \hat{\xi}_i = \frac{1}{\bar{\rho}} + \frac{D}{\bar{\beta}\bar{\rho}} \sum_{i=1}^{m} \bar{\xi}_i = \frac{1}{\bar{\beta}} = \sum_{i=1}^{m} \hat{u}_i$. A similar argument holds for the second assertion. □

Practically we found ν-LP (4) with $D = \frac{1}{m\nu}$, $\nu \in (0, 1)$ preferable because of the interpretability of the parameter (see (Ratsch et al., 2000)). To maintain dual feasibility, the parameter ν must maintain $\frac{1}{m} <= D <= 1$. By picking ν we can force the minimum number of support vectors. The number of support vectors will be the number of points misclassified plus the points on the margin, and thus can be used as a heuristic for choice of ν.

LP (4) has a very interesting interpretation. The dual LP assigns classification costs u_i to each point such that the v_i sum to 1. The constraint $\sum_{i=1}^{m} u_i y_i H_{ij} \leq \beta$ "scores" each weak learner h_j. The score is the weighted sum of the correctly classified points minus the weighted sum of the incorrectly classified points. The set of best weak learners has a score of β. If the class of weak learners is closed under complementation, we can safely assume that $\beta > 0$ since any weak learner with negative cost has a complementary weak learner with positive cost. The objective minimizes β so the optimal classification cost u is the most pessimistic one, i.e. it minimizes the maximum score on all the weak learners. From complementary slackness, only the weak learners with scores equal to β can have positive weights a_j in the primal space. The optimal ensemble is a linear combination of the weak learners that perform best under the most pessimistic choice of classification costs. This interpretation closely corresponds to the game strategy approach of (Breiman,

1999) (also an LP solvable by LPBoost.) A notable difference is that LP (5) has an upper bound on the classification costs u that is produced by the introduction of the soft margin in the primal.

4. Column Generation Algorithm

Since the matrix H has a very large number of columns, prior authors have dismissed the idea of solving LP formulations for boosting as being intractable using standard LP techniques. But column generation (CG) techniques for solving such LPs have existed since the 1950's and can be found in LP text books see for example (Nash & Sofer, 1996, Section 7.4). The idea of CG is to restrict the primal problem (2) by considering only a subset of all the possible labelings based on the weak learners generated so far; i.e., only a subset $\hat{H}$ of the columns of H is used. The LP solved using $\hat{H}$ is typically refered to as the *restricted master problem*. Solving the restricted primal LP corresponds to solving a relaxation of the dual LP. The constraints for weak learners that have not been generated yet are missing. One extreme case is when no weak learners are considered. In this case the optimal dual solution is $\hat{u}_i = \frac{1}{m}$ (with appropriate choice of D). This will provide the initialization of the algorithm.

If we consider the unused columns to have $\hat{a}_i = 0$, then $\hat{a}$ is feasible for the original primal LP. If $(\hat{u}, \hat{\beta})$ is feasible for the original dual problem then we are done since we have primal and dual feasibility with equal objectives. If $\hat{a}$ is not optimal then $(\hat{u}, \hat{\beta})$ is infeasible for the dual LP with full matrix H. Specifically, the constraint $\sum_{i=1}^m \hat{u}_i y_i H_{ij} \leq \hat{\beta}$ is violated for at least one weak learner. Or equivalently, $\sum_{i=1}^m \hat{u}_i y_i H_{ij} > \hat{\beta}$ for some j. Of course we don't want to a priori generate all columns of H $(H_{.j})$, so we use our weak learner as an oracle that either produces $H.j$, $\sum_{i=1}^m \hat{u}_i y_i H_{ij} > \hat{\beta}$ for some j or a guarantee that no such $H_{.j}$ exists. To speed convergence we would like to find the one with maximum deviation, that is, the weak learning algorithm $\mathcal{H}(S, u)$ must deliver a function $\hat{h}$ satisfying

$$\sum_{i=1}^m y_i \hat{h}(x_i)\hat{u}_i = \max_{h \in \mathcal{H}} \sum_{i=1}^m \hat{u}_i y_i h(x_i) \qquad (6)$$

Thus $\hat{u}_i$ becomes the new misclassification cost for example i that is given to the weak learning machine to guide the choice of the next weak learner. One of the big pay offs of the approach is that we have a stopping criterion. If there is no weak learner h for which $\sum_{i=1}^m \hat{u}_i y_i h(x_i) > \hat{\beta}$, then the current combined hypothesis is the optimal solution over all linear combinations of weak learners.

We can gauge the cost of early stopping since if $\max_{h \in \mathcal{H}} \sum_{i=1}^m \hat{u}_i y_i h(x_i)) \leq \hat{\beta} + \epsilon$, for some $\epsilon > 0$, we can obtain a feasible solution of the full dual problem by taking $(\hat{u}, \hat{\beta} + \epsilon)$. The value V of the optimal solution can be bounded between $\hat{\beta} \leq V < \hat{\beta} + \epsilon$. This implies that, even if we were to potentially include a non-zero coefficient for all the weak learners, the value of the objective $\rho - D \sum_{i=1}^m \xi_i$ can only be increased by at most ϵ.

We assume the existence of the weak learning algorithm $\mathcal{H}(S, u)$ which selects the best weak learner from a set H closed under complementation using the criterion of equation (6). The following algorithm results

Algorithm 4.1 (LPBoost).

Given as input training set: S
$n \leftarrow 0$ *No weak learners*
$a \leftarrow 0$ *All coefficients are* 0
$\beta \leftarrow 0$
$u \leftarrow (\frac{1}{m}, \ldots, \frac{1}{m})$ *Corresponding optimal dual*
REPEAT
  $n \leftarrow n + 1$
  Find weak learner using equation (6):
  $h_n \leftarrow \mathcal{H}(S, u)$
  Check for optimal solution:
  $If \sum_{i=1}^m u_i y_i h_n(x_i) \leq \beta, n \leftarrow n - 1,$ *break*
  $H_{in} \leftarrow h_n(x_i)$
  Solve restricted master for new costs:

$$(u, \beta) \leftarrow \begin{array}{ll} argmin & \beta \\ s.t. & \sum_{i=1}^m u_i y_i h_j(x_i) \leq \beta \\ & j = 1, \ldots, n \\ & 0 \leq u_i \leq D, \; i = 1, \ldots, m \end{array}$$

END
$a \leftarrow$ *Lagrangian multipliers from last LP*
return $n, f = \sum_{j=1}^n a_j h_j$

With small changes this algorithm can be adapted to perform any of the LP boosting formulations by simply changing the restricted master LP solved and the optimality conditions checked. Assuming the base learner solves (6) exactly, LPBoost is a dual simplex algorithm (Nash & Sofer, 1996). Thus it inherits all the benefits of the simplex algorithm. Benefits include: 1) Well defined exact and approximate stopping criteria. Typically ad hoc termination schemes, e.g. a fixed number of iterations, must be used for the gradient-based boosting algorithms. 2) Finite termination at a globally optimal solution. In practice the algorithm generates few weak learners to arrive at an optimal solution. 3) The optimal solution is sparse and thus uses few weak learners. 4) The algorithm is performed in the dual space of the classification costs. The weights of the optimal ensemble are only generated and fixed

Table 1. Average Accuracy of Boosting using Stumps: # of weak learners = (n) for LPBoost, and 100, 1000 for AdaBoost.

Dataset	LPBoost (n)	AB-100	AB-1000
Cancer	0.9657 (14.7)	0.9542	0.9471
Heart	0.7946 (70.8)	0.8182	0.8014
Sonar	0.8702 (85.7)	0.8077	0.8558
Iono	0.9060 (87.6)	0.9060	0.9031
Diag	0.9613 (54.2)	0.9684	0.9701
Musk	0.8824 (205.3)	0.8403	0.8908

at optimality. 5) High-performance commercial LP algorithms optimized for column generation exist that do not suffer from the numeric instability problems reported for boosting (Bauer & Kohavi, 1999). It is not necessary that the base leaner solve (6) exactly. All that is needed for convergence (assuming no problems with cycling) is that hypothesis h_n satisfying $\sum_{i=1}^{m} u_i y_i h_n(x_i) > \beta$ be generated. If it fails to do this when such a hypothesis exists then the algorithm may terminate prematurely.

5. Computational Experiments

Two sets of experiments to compare the performance of LPBoost and AdaBoost were performed: one boosting decision tree stumps on smaller datasets and one boosting C4.5 (Quinlan, 1996) on large datasets with noise. The rationale was to first evaluate LPBoost where the base learner solves (6) exactly, then to examine LPBoost in a more realistic environment.

5.1 Boosting Decision Tree Stumps

We used decision tree stumps as a base learner on six UCI datasets: Cancer (9,699), Heart (13,297), Sonar (60,208), Ionosphere (34,351), Diagnostic (30,569), and Musk (166,476). The number of features and number of points in each dataset are shown in parentheses respectively. We report testing set accuracy for each dataset based on 10-fold Cross Validation (CV). We generate the decision stumps based on the mid-point between two consecutive points for a given variable. Both LPBoost and AdaBoost search for the best weak learner which returns the least weighted misclassification error at each iteration. LPBoost takes advantage of the fact that each weak learner need only be added once since the weight on the weak learner may be changed. Thus once a decision stump is added to the ensemble it is no longer considered. Adaboost considers all decision stumps at each iteration. The parameter ν for LPboost was set using a simple heuristic: 0.1 added to previously reported error rates on each dataset in (Bennett & Demiriz, 1999) except for the Cancer dataset. Specifically the values of ν in the same order of the datasets given above were (0.2, 0.25, 0.3, 0.2, 0.1, 0.25). Results for Adaboost were reported for maximum of iterations of 100 and 1000. The 10-fold average accuracies are reported in Table 1.

LPBoost performed very well both in terms of classification accuracy, number of weak learners, and training time. There is little difference between the accuracy of LPBoost and the best accuracy reported for AdaBoost using either 100 or 1000 iterations. There was only a significant difference between LPBoost and the best AdaBoost accuracy (based on p-value of 0.1) on the Heart dataset. The variation in AdaBoost for 100 and 1000 iterations illustrates the importance of well defined stopping criteria. Typically, AdaBoost only obtains its solution in the limit and thus stops when the maximum number of iterations (or some other heuristic stopping criteria) is reached. There is no magic number of iterations good for all datasets. LPBoost has a well defined stopping criterion that is reached in a few iterations. It uses few weak learners. There are only 81 possible stumps on the Breast Cancer dataset (9 attributes having 9 possible values), so clearly AdaBoost may require the same tree to be generated multiple times. LPBoost generates a weak learner only once and can alter the weight on that weak learner at any iteration. The run time of LPBoost is proportional to the number of weak learners generated. Since the LP package that we used, CPLEX 4.0 (CPLEX, 1994), is optimized for column generation, the cost of adding a column and reoptimizing the LP at each iteration is small. An iteration of LPBoost is only slightly more expensive than an iteration of AdaBoost. The time is proportional to the number of weak learners generated. For problems where LPBoost generates far fewer weak learners it is much less computationally costly.

In the next subsection, we test the practicality of our methodology on larger datasets.

5.2 Boosting C4.5

LPBoost with C4.5 performed well on large datasets after some operational challenges were solved. In concept boosting using C4.5 is straight forward since the C4.5 algorithm accepts classification costs. But C4.5 only finds a good solution not guaranteed to maximize (6). Another challenge is that the classification costs determined by LPBoost are sparse, i.e. $u_i = 0$ for many of the points. The dual LP has a basic feasible solution corresponding to a vertex of the dual feasible region. Only the variables corresponding to the basic solution can be nonnegative. So while a face of the region corresponding to many nonnegative weights may be optimal, only a vertex solution will be chosen. With many $u_i = 0$, the c4.5 solutions based on data subsets were not very helpful over the full data, thus LPBoost made little improvement in 25 iterations over the first equal cost solution. Other authors have reported problems with underflow of boosting (Bauer & Kohavi, 1999). When LPBoost was solved to optimality on decision stumps with full evaluation of the weak learners, this problem did not occur.

Stability was greatly improved by adding minimum classification weights to the dual LP (5) :

$$\begin{array}{ll} \min_u & \beta \\ s.t. & \sum_{i=1}^m u_i y_i H_{ij} \le \beta,\ j = 1, \dots, n \\ & \sum_{i=1}^m u_i = 1 \\ & D' \le u_i \le D,\ i = 1, \dots, m \end{array} \quad (7)$$

where $D = \frac{1}{\nu m}$ and $D' = \frac{1}{25 \nu m}$. The corresponding primal problem is

$$\begin{array}{ll} \max_{a,\xi,\rho} & \rho + D' \sum_{i=1}^m \tau_i - D \sum_{i=1}^m \xi_i \\ s.t. & y_i H_i a + \xi_i \ge \rho + \tau_i,\ i = 1, \dots, m \\ & \sum_{j=1}^n a_j = 1,\ \ , a_j \ge 0,\ j = 1, \dots, n \\ & \xi_i \ge 0,\ i = 1, \dots, m \end{array} \quad (8)$$

The primal problem maximizes two measures of soft margin: ρ corresponds to the minimum margin obtained by all points and τ_i measures the additional margin obtained by each point. AdaBoost also minimizes a margin cost function based on the margin obtained by each point.

We ran experiments on two larger datasets: Forest and Adult, from UCI(Murphy & Aha, 1992). Forest is a 54-dimension dataset with 7 possible classes. The data are divided into 11340 training, 3780 validation and 565892 testing instances. There are no missing values. LPBoost was adopted to the multiclass by defining $h_j(x_i) = 1$ if instance x_i is correctly classified by weak learner h_j and -1 otherwise. This is only one way to address the multiclass problem and further investigation is needed to determine if it is the most appropriate. The 15-dimensional Adult dataset has 32562 training and 16283 testing instances. One training point which has a missing value for a class label has been removed. We use 8140 instances as our training set and the remaining 24421 instances as the validation set. Adult is a two-class dataset with missing values. The default handling in C4.5 has been used for missing values. In order to investigate the performance of boosted C4.5 with noisy data, we introduced 15% label noise for both the Forest and Adult datasets.

The ν parameter used in LPBoost and the number of iterations of AdaBoost can significantly affect their

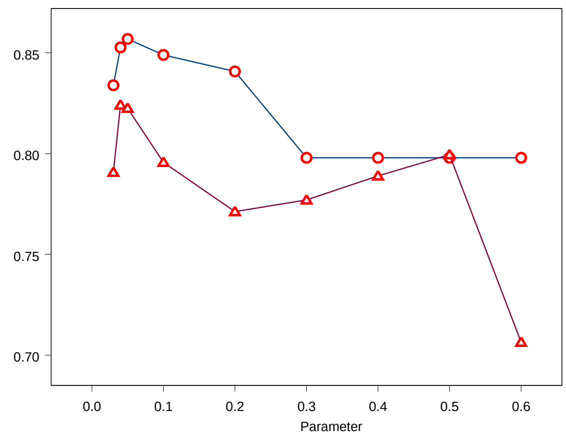

Figure 1. Forest Validation Set Accuracy by ν Value *Circles are no noise and triangles are with noise.*

Table 2. Results from Boosting C4.5

Dataset	LPBoost	AdaBoost	C4.5
Original Forest	0.7320	0.7398	0.6638
+ 15% Noise	0.6934	0.6883	0.5927
Original Adult	0.8503	0.8371	0.8288
+ 15% Noise	0.8121	0.7744	0.7625

performance. Thus accuracy on the validation set was used to pick the parameter ν for LPBoost and the number of iterations. Due to the excessive computational work, we limit the maximum number of iterations at 25 for both LPBoost and AdaBoost as in (Bauer & Kohavi, 1999). AdaBoost used 5 weak learners on the Adult dataset with 15% noise and 23 weak learners on the Forest dataset with 15% noise. On all other runs both algorithms terminated at 25 weak learners. Figure 1 shows the validation accuracy for LPBoost on the Forest. The testing set results for ν with the best validation set accuracy are given in Table 2 ($\nu = .05$ for all datasets except $\nu = .04$ on Forest with noise). LPBoost was very comparable with AdaBoost in terms of CPU time. As seen in Table 2, LPBoost is perfroms very well in comparison with AdaBoost when the validation set is used to pick the best parameter settings. Computational time is dominated by the C4.5 costs . The reoptimization cost of LPBoost at each iteration is very small. On an IBM RS-6000, the CPU times in seconds for 25 boosting iterations were LPBoost: 89 and AdaBoost: 107 for Adult and LPBoost: 930 and AdaBoost: 717 for Forest.

6. Discussion and Extensions

We have shown that LP formulations of boosting are both attractive theoretically in terms of generalization error bound and computationally via column generation. We showed the equivalence of our original soft margin formulation and the ν formulation (Ratsch et al., 2000) and found an interpretation closely related to (Breiman, 1999). The LPBoost algorithm can be applied to any boosting problem formulated as an LP. With modification of the original LP, the method performed very well in practice on large datasets. From an optimization perspective, LPBoost has many benefits over gradient-based approaches: finite termination, numerical stability, well-defined convergence criteria, very fast algorithms in practice, and fewer weak learners in the optimal ensemble. LPBoost may be more sensitive to inexactness of the base learning algorithm. But through modification of the base LP, we were able to obtain very good performance in large noisy datasets. The question of what is the best LP formulation for boosting remains open. Confidence-rated boosting using the probability of correctness of the prediction may lead to improvements. But clearly LP formulations are tractable using column generation, and should be the subject of further research.

Acknowledgements

This material is based on research supported by Microsoft Research, NSF Grants 949427 and IIS-9979860, and the European Commision under the Working Group Nr. 27150 (NeuroCOLT2).

References

Anthony, M., & Bartlett, P. (1999). *Learning in neural networks : Theoretical foundations*. Cambridge University Press.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*, 105–139.

Bennett, K. P. (1999). Combining support vector and mathematical programming methods for classification. *Advances in Kernel Methods – Support Vector Machines* (pp. 307–326). Cambridge, MA: MIT Press.

Bennett, K. P., & Demiriz, A. (1999). Semi-supervised support vector machines. *Advances in Neural Information Processing Systems 11* (pp. 368–374). Cambridge, MA: MIT Press.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, *11*, 1493–1517.

CPLEX (1994). *Using the CPLEX callable library*. CPLEX Optimization Incorporated, Incline Village, Nevada.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press.

Mangasarian, O. L. (2000). Generalized support vector machines. *Advances in Large Margin Classifiers* (pp. 135–146). Cambridge, MA: MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

Murphy, P., & Aha, D. (1992). *UCI repository of machine learning databases*. Department of Information and Computer Science, University of California, Irvine, California.

Nash, S., & Sofer, A. (1996). *Linear and nonlinear programming*. New York, NY: McGraw-Hill.

Quinlan, J. (1996). Bagging, boosting, and C4.5. *Proceedings of the 13th National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Rätsch, G., Schölkopf, B., Smola, A., Mika, S., Onoda, T., & M uller, K.-R. (1999). Robust ensemble learning. *Advances in Large Margin Classifiers* (pp. 208–222). Cambridge, MA: MIT Press.

Rätsch, G., Schölkopf, B., Smola, A., M uller, K.-R., Onoda, T., & Mika, S. (2000). ν-arc ensemble learning in the presence of outliers. *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press.

Schapire, R., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, *26*, 1651–1686.

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

Shawe-Taylor, J., & Cristianini, N. (1999). Margin distribution bounds on generalization. *Proceedings of the European Conference on Computational Learning Theory, EuroCOLT'99* (pp. 263–273).

Zhang, T. (1999). *Analysis of regularised linear functions for classification problems* (Technical Report RC-21572). IBM.