Linear Programming Boosting via Column Generation

Ayhan Demiriz (demira@rpi.edu)
Dept. of Decision Sciences and Eng. Systems, Rensselaer Polytechnic Institute, Troy, NY 12180 USA

Kristin P. Bennett (bennek@rpi.edu)
Dept. of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180 USA while visiting Microsoft Research, Redmond, WA USA

John Shawe-Taylor (jst@cs.rhul.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 based simplex method. We formulate the problem as if all possible weak hypotheses had already been generated. The labels produced by the weak hypotheses become the new feature space of the problem. The boosting task becomes to construct a learning function in the label space that minimizes misclassification error and maximizes the soft margin. We prove that for classification, minimizing the 1-norm soft margin error function directly optimizes a generalization error bound. The equivalent linear program can be efficiently solved using column generation techniques developed for large-scale optimization problems. The resulting LPBoost algorithm can be used to solve any LP boosting formulation by iteratively optimizing the dual misclassification costs in a restricted LP and dynamically generating weak hypotheses to make new LP columns. We provide algorithms for soft margin classification, confidence-rated, and regression boosting problems. Unlike gradient boosting algorithms, which may converge in the limit only, LPBoost converges in a finite number of iterations to a global solution satisfying mathematically well-defined optimality conditions. The optimal solutions of LPBoost are very sparse in contrast with gradient based methods. Computationally, LPBoost is competitive in quality and computational cost to AdaBoost.

Keywords: Ensemble Learning, Boosting, Linear Programming, Sparseness, Soft Margin

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 (Rätsch et al., 2000a; Breiman, 1999).

The main contribution in the application of LP methods to boosting has been made by Grove and Schuurmans (Grove & Schuurmans, 1998) who derived an LP method DualLPBoost based on maximizing the margin of the combined classifier. They experienced difficulties, however, in getting the method to work in practice. We adopt a similar approach but optimize a new rigorous generalization bound obtained in terms of a soft margin measure. Using a soft margin ensures that the approach is able to handle noisy data more robustly, but also overcomes the convergence problems experienced by Grove and Schuurmans. We discuss in more detail the reasons for this improvement in Section 7. Furthermore in this paper we show that LP boosting is generally 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 (Rätsch et al., 2000b; Bennett, 1999; Mangasarian, 2000). One advantage of these approaches is that immediately the method of analysis for support vector machine problems becomes applicable to the boosting problem. In Section 2, we prove that the LPBoost approach to classification directly minimizes a bound on the generalization error. We adopt the LP formulations developed for support vector machines. In Section 3, we discuss the soft margin LP formulation. By adopting linear programming, we immediately have the tools of mathematical programming at our disposal. In Section 4 we examine how column generation approaches for solving large scale LPs can be adapted to boosting.

For classification, we examine both standard and confidence-rated boosting. Standard boosting algorithms use weak hypotheses that are classifiers, that is, whose outputs are in the set $\{-1, +1\}$. Schapire and Singer (Schapire & Singer, 1998) have considered boosting weak hypotheses whose outputs reflected not only a classification but also an associated confidence encoded by a value in the range $[-1, +1]$. They demonstrate that so-called confidence-rated boosting can speed convergence of the composite classifier, though the accuracy in the long term was not found to be significantly affected. In Section 5, we discuss the minor modifications needed for LPBoost to perform confidence-rated boosting.

The methods we develop can be readily extended to any ensemble problem formulated as an LP. We demonstrate this by adapting the approach to regression in Section 6. In Section 7, we examine the hard margin LP formulation of (Grove & Schuurmans, 1998) which is also a special case of the column generation approach. By use of duality theory and optimality conditions, we can gain insight into how LP boosting works mathematically, specifically demonstrating the critical differences between the prior hard margin approach and the proposed soft margin approach. Computational results and practical issues for implementation of the method are given in Section 8.

2. Motivation for Soft Margin Boosting

We begin with an analysis of the boosting problem using the methodology developed for support vector machines. The function classes that we will be considering are of the form $\mathrm{co}(H) = \{\sum_{h \in H} a_h h : a_h \geq 0\}$, where H is a set of weak hypotheses 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 $\mathrm{err}_{\mathcal{D}}(f)$ 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 $S = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell)$ is a finite set of functions A such that for all $f \in \mathcal{F}$, there exists $g \in A$, such that $\max_{1 \leq i \leq \ell} (|f(\mathbf{x}_i) - g(\mathbf{x}_i)|) < \gamma$. 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*

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

In the remainder of this section we will assume a training set $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell))$. 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 $m_S(f) = \min_{1 \leq i \leq \ell} (y_i f(\mathbf{x}_i))$.

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 it 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* ℓ *random examples* S, *any hypothesis* $f \in \mathcal{F}$ *that has margin* $m_S(f) \geq \gamma$ *on* S *has error no more than*

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

provided $\ell > 2/\varepsilon$.

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*

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

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 the 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* $\xi((\mathbf{x}_i, y_i), f, \gamma) = \xi_i = \max(0, \gamma - y_i f(\mathbf{x}_i))$. *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

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

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* ℓ *random examples* S, *any hypothesis* $f \in \mathcal{F}$ *for which* $(f, g_f) \in \mathcal{G}$ *has generalization error no more than*

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

provided $\ell > 2/\varepsilon$, *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, $\mathcal{F} = \mathrm{co}(H) = \{\sum_{h \in H} a_h h : a_h \geq 0\}$, where we have left open for the time being what the class H of weak hypotheses 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

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

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 hypotheses H, the bound B, and the margin γ. Before launching into this analysis, observe that for any input $\mathbf{x}$,

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

2.1. Covering Numbers of Convex Hulls

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

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

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 $S = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell)$ 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 $G \cup \{1, \ldots, \ell\}$. Let $\mathcal{V}_B$ be the function class $\mathcal{V}_B = \{\mathbf{g} \mapsto \langle \mathbf{g} \cdot v\rangle : v \in V, \|v\|_1 \le B, \|\mathbf{g}\|_\infty \le 1\}$, and suppose that U is an $(\gamma - \eta)$-cover of $\mathcal{V}_B$. We claim that the set

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

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 $f = (\sum_{h\in H} a_h h, g) \in \mathcal{G}_B$, 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 $\left|h(\mathbf{x}_i) - \hat{h}(\mathbf{x}_i)\right| \le \eta/B$, and for $h' \in G$ set $v_{h'} = \sum_{h:\hat{h}=h'} a_h$ and $v_i = g(\mathbf{x}_i)/\Delta$, $i = 1, \ldots, \ell$. Now we form the function $\bar{f} = \left(\sum_{h\in G} v_h h, \sum_{i=1}^{\ell} v_i \delta_{\mathbf{x}_i}\right)$, which lies in the set $\mathcal{V}_B$, since $\sum_{h \in G} a_h + \sum_{i=1}^{\ell} v_i \leq B$. Furthermore we have that

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

Since U is a $\gamma - \eta$ cover of $\mathcal{V}_B$ there exists $\hat{v} \in U$ such that $\hat{f} = \left(\sum_{h \in G} \hat{v}_h h, \sum_{i=1}^{\ell} \hat{v}_i \delta_{\mathbf{x}_i}\right)$ is within $\gamma - \eta$ of $\bar{f}$ on $\tau_\Delta(\mathbf{x}_j)$, $j = 1, \ldots, \ell$. 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*

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

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 $B_H(\ell)$ of the class H of weak hypotheses.

3. Boosting LP for Classification

From the above discussion we can see that a soft margin cost function should be valuable for boosting classification functions. Once again using the techniques used in support vector machines, we can formulate this problem as a linear program. 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 hypotheses 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 hypotheses are generated as needed to produce the optimal support vector machine based on the output of the all weak hypotheses. In essence the base learning algorithm becomes an 'oracle' that generates the necessary columns. The dual variables of the linear program provide the misclassification costs needed by the learning machine. The column generation procedure searches for the best possible misclassification costs in dual space. Only at optimality is the actual ensemble of weak hypotheses constructed.

3.1. LP Formulation

Let the matrix H be a ℓ by m matrix of all the possible 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 hypothesis $h_j \in \mathcal{H}$ on the training point x_i. Each column $H_{.j}$ of the matrix H constitutes the output of weak hypothesis h_j on the training data, while each row H_i gives the outputs of all the weak hypotheses on the example x_i. There may be up to 2^ℓ distinct weak hypotheses.

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

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

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

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

Alternative soft margin LP formulations exist, such as this one for the ν-LP Boosting[1] (Rätsch et al., 2000a):

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

The dual of this LP (4) is:

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

[1] We remove the constraint $\rho \geq 0$ since $\rho > 0$ provided D is sufficiently small.

These LP formulations are exactly equivalent given the appropriate choice of the parameters C and D. Proofs of this fact can be found in (Rätsch et al., 2000b; Bennett et al., 2000) so we only state the theorem here.

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* $(\bar{u}, \bar{\beta})$*, 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* $(\hat{u} \neq 0)$*, then* $(\bar{\rho} = \frac{1}{\sum_{i=1}^{m} \hat{a}_i}, \bar{a} = \hat{a}\bar{\rho}, \bar{\xi} = \hat{\xi}\bar{\rho})$ *and* $(\bar{\beta} = \frac{1}{\sum_{i=1}^{\ell} \hat{u}_i}, \bar{u} = \hat{u}\bar{\beta})$ *are the primal and dual solutions of LP (4) with parameter* $D = C\hat{\beta}$*.*

Practically we found ν-LP (4) with $D = \frac{1}{\ell\nu}$, $\nu \in (0, 1)$ preferable because of the interpretability of the parameter. A more extensive discussion and development of these characteristics for SVM classification can be found in (Rätsch et al., 2000b). To maintain dual feasibility, the parameter ν must maintain $\frac{1}{\ell} <= D <= 1$. By picking ν appropriately we can force the minimum number of support vectors. We know that the number of support vectors will be the number of points misclassified plus the points on the margin, and this was used as a heuristic for choice of ν. The reader should consult (Rätsch et al., 2000a, 2000b) for a more in-depth analysis of this family of cost functions.

3.2. Properties of LP formulation

We now examine the characteristics of LP (4) and its optimality conditions to gain insight into the properties of LP Boosting. This will be useful in understanding both the effects of the choice of parameters in the model and the performance of the eventual algorithm. The optimality conditions (Nash & Sofer, 1996) of LP (4) are *primal feasibility*:

$$\begin{array}{l} y_i H_i a + \xi_i \geq \rho, \ i = 1, \dots, \ell \\ \sum_{i=1}^{m} a_i = 1, \quad \xi_i \geq 0, \ i = 1, \dots, \ell \\ a_i \geq 0, \ i = 1, \dots, m \end{array} \tag{6}$$

dual feasibility:

$$\begin{array}{l} \sum_{i=1}^{\ell} u_i y_i H_{ij} \leq \beta, \ j = 1, \dots, m \\ \sum_{i=1}^{\ell} u_i = 1, \quad 0 \leq u_i \leq D, \ i = 1, \dots, \ell \end{array} \tag{7}$$

and *complementarity*, here stated as equality of the primal and dual objectives:

$$\rho - D\sum_{i=1}^{\ell} \xi_i = \beta \tag{8}$$

Complementarity can be expressed using many equivalent formulations. For example, from the complementarity property, the following equations hold:

$$\begin{array}{l} u_i(y_i H_i a + \xi_i - \rho) = 0, \ i = 1, \ldots, \ell \\ a_j(\sum_{i=1}^{\ell} u_i y_i H_{ij} - \beta) = 0, \ j = 1, \ldots, m \end{array} \tag{9}$$

As in SVM, the optimality conditions tell us many things. First we can characterize the set of base hypotheses that are positively weighted in the optimal ensemble. Recall that the primal variables a_i multiply each base hypothesis. The dual LP assigns misclassification costs u_i to each point such that the u_i sum to 1. The dual constraint $\sum_{i=1}^{\ell} u_i y_i H_{ij} \le \beta$ "scores" each weak hypothesis h_j. The score is the weighted sum of the correctly classified points minus the weighted sum of the incorrectly classified points. The weak hypotheses with lower scores have greater weighted misclassification costs. The formulation is pessimistic in some sense. The set of best weak hypotheses for a given u will all have a score of β. The dual objective minimizes β so the optimal misclassification cost u will be the most pessimistic one, i.e., it minimizes the maximum score over all the weak hypotheses. From the complementary slackness condition, $a_j(\sum_{i=1}^{\ell} u_i y_i H_{ij} - \beta) = 0, \ j = 1, \ldots, m$, only the weak hypotheses with scores equal to β can have positive weights a_j in the primal space. So the resulting ensemble will be a linear combination of the weak hypotheses that perform best under the most pessimistic choice of misclassification costs. This interpretation closely corresponds to the game strategy approach of (Breiman, 1999) (which is also a LP boosting formulation solvable by LPBoost.) A notable difference is that LP (5) has an additional upper bound on the misclassification costs u, $0 \le u_i \le D, \ i = 1, \ldots, \ell$, that is produced by the introduction of the soft margin in the primal.

From the LP optimality conditions and the fact that linear programs have extreme point solutions, we know that there exist very sparse solutions of both the primal and dual problems and that the degree of sparsity will be greatly influenced by the choice of parameter $D = \frac{1}{\nu\ell}$. The size of the dual feasible region depends on our choice of ν. If ν is too large, forcing D small, then the dual problem is infeasible. For large but still feasible ν (D very small but still feasible), the problem degrades to something very close to the equal-cost case, $u_i = 1/\ell$.

All the u_i are forced to be nonzero. Practically, this means that as ν increases (D becomes larger), the optimal solution may be one or two weak hypotheses that are best assuming approximately equal costs. As ν decreases (D grows), the misclassification costs, u_i, will increase for hard-to-classify points or points on the margin in the label space and will go to 0 for points that are easy to classify. Thus the misclassification costs u become sparser. If ν is too small (and D too large) then the meaningless null solution, $a = 0$, with all points classified as one class, becomes optimal.

For a good choice of ν, a sparse solution for the primal ensemble weights a will be optimal. This implies that few weak hypotheses will be used. Also a sparse dual u will be optimal. This means that the solution will be dependent only on a smaller subset of data (the support vectors.) Data with $u_i = 0$ are well-classified with sufficient margin, so the performance on these data is not critical. From LP sensitivity analysis, we know that the u_i are exactly the sensitivity of the optimal solution to small perturbations in the margin. In some sense the sparseness of u is good because the weak hypotheses can be constructed using only smaller subsets of the data. But as we will see in Sections 7 and 8, this sparseness of the misclassification costs can lead to problems when implementing algorithms.

4. LPBoost Algorithms

We now examine practical algorithms for solving the LP (4). 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 techniques for solving such LPs have existed since the 1950s and can be found in LP textbooks; see for example (Nash & Sofer, 1996, Section 7.4). Column generation is frequently used in large-scale integer and linear programming algorithms so commercial codes such as CPLEX have been optimized to perform column generation very efficiently (CPLEX, 1994). The simplex method does not require that the matrix H be explicitly available. At each iteration, only a subset of the columns is used to determine the current solution (called a basic feasible solution). The simplex method needs some means for determining if the current solution is optimal, and if it is not, some means for generating some column that violates the optimality conditions. The tasks of verification of optimality and generating a column can be performed by the learning algorithm. A simplex-based boosting method will alternate between solving an LP for a reduced matrix $\hat{H}$ corresponding to the weak hypotheses generated so far and using the base learning algorithm to generate the best-scoring weak hypothesis based on the dual misclassification cost provided by the LP. This will continue until the algorithm terminates at an exact or approximate optimal solution based on well-defined stopping criteria or some other stopping criteria such as the maximum number of iterations is reached.

The idea of column generation (CG) is to restrict the primal problem (2) by considering only a subset of all the possible labelings based on the weak hypotheses 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 referred to as the *restricted master problem.* Solving the restricted primal LP corresponds to solving a relaxation of the dual LP. The constraints for weak hypotheses that have not been generated yet are missing. One extreme case is when no weak hypotheses are considered. In this case the optimal dual solution is $\hat{u}_i = \frac{1}{\ell}$ (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 $\sum_{i=1}^{\ell} \hat{u}_i y_i H_{ij} \leq \hat{\beta}$ is violated for at least one weak hypothesis. Or equivalently, $\sum_{i=1}^{\ell} \hat{u}_i y_i H_{ij} > \hat{\beta}$ for some j. Of course we do not want to a priori generate all columns of H $(H_{.j})$, so we use our base learning algorithm as an oracle that either produces $H_{.j}$, $\sum_{i=1}^{\ell} \hat{u}_i y_i H_{ij} > \hat{\beta}$ 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 base learning algorithm $\mathcal{H}(S, u)$ must deliver a function $\hat{h}$ satisfying

$$\sum_{i=1}^{\ell} y_i \hat{h}(x_i) \hat{u}_i = \max_{h \in \mathcal{H}} \sum_{i=1}^{\ell} \hat{u}_i y_i h(x_i) \tag{10}$$

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

We can also gauge the cost of early stopping since if $\max_{h \in \mathcal{H}} \sum_{i=1}^{\ell} \hat{u}_i y_i h(x_i)) \leq \hat{\beta} + \epsilon$, for some $\epsilon > 0$, we can obtain a feasible solution of the full dual problem by taking $(\hat{u}, \hat{\beta} + \epsilon)$. Hence, 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 hypotheses, the value of the objective $\rho - D\sum_{i=1}^{\ell}\xi_i$ 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 hypothesis from a set H closed under complementation using the criterion of equation (10). The following algorithm results.

Algorithm 4.1 (LPBoost).

Given as input training set: S
$m \leftarrow 0$ *No weak hypotheses*
$a \leftarrow 0$ *All coefficients are* 0
$\beta \leftarrow 0$
$u \leftarrow (\frac{1}{\ell},\ldots,\frac{1}{\ell})$ *Corresponding optimal dual*
REPEAT
  $m \leftarrow m+1$
  Find weak hypothesis using equation (10) :
  $h_m \leftarrow \mathcal{H}(S,u)$
  Check for optimal solution:
  If $\sum_{i=1}^{\ell} u_i y_i h_m(x_i) \le \beta, m \leftarrow m-1,$ *break*
  $H_{im} \leftarrow h_m(x_i)$
  Solve restricted master for new costs:

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

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

Note that the assumption of finding the best weak hypothesis is not essential for good performance of the algorithm. Recall that the role of the learning algorithm is to generate columns (weak hypotheses) corresponding to a dual infeasible row or to indicate optimality by showing no infeasible weak hypotheses exist. All that we require is that the base learner return a column corresponding to a dual infeasible row. It need not be the one with maximum infeasibility. This is done primarily to improve convergence speed. In fact, choosing columns using "steepest edge" criteria that look for the column that leads to the biggest actual change in the objective may lead to even faster convergence. If the learning algorithm fails to find a dual infeasible weak hypothesis when one exists then the algorithm may prematurely stop at a nonoptimal solution.

With small changes this algorithm can be adapted to perform any of the LP boosting formulations by simply changing the restricted master LP solved, the costs given to the learning algorithm, and the optimality conditions checked. Assuming the base learner solves (10) exactly, LPBoost is a variant of the 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 for global optimality. Typically, ad hoc termination schemes, e.g. a fixed number of iterations, are the only effective termination criteria for the gradient-based boosting algorithms. 2) Finite termination at a globally optimal solution. In practice the algorithm generates few weak hypotheses to arrive at an optimal solution. 3) The optimal solution is sparse and thus uses few weak hypotheses. 4) The algorithm is performed in the dual space of the classification costs. The weights of the optimal ensemble are only generated and fixed at optimality. 5) High-performance commercial LP algorithms optimized for column generation exist making the algorithm efficient in practice.

5. Confidence-rated Boosting

The derivations and algorithm of the last two sections did not rely on the assumption that $L_{ij} \in \{-1, +1\}$. We can therefore apply the same reasoning to implementing a weak learning algorithm for a finite set of confidence-rated functions $\mathcal{F}$ whose outputs are real numbers. We again assume that $\mathcal{F}$ is closed under complementation. We simply define $L_{ij} = f_j(x_i)$ for each $f_j \in \mathcal{F}$ and apply the same algorithm as before. We again assume the existence of a base learner $F(S, u)$, which finds a function $\hat{f} \in \mathcal{F}$ satisfying

$$\sum_{i=1}^{\ell} y_i \hat{f}(x_i) \hat{u}_i = \max_{f \in \mathcal{F}} \sum_{i=1}^{\ell} \hat{u}_i y_i f(x_i) \tag{11}$$

The only difference in the associated algorithm is the base learner which now optimizes this equation.

Algorithm 5.1 (LPBoost-CRB).

Given as input training set: S
$m \leftarrow 0$ *No weak hypotheses*
$a \leftarrow 0$ *All coefficients are* 0
$\beta \leftarrow 0$
$u \leftarrow (\frac{1}{\ell}, \ldots, \frac{1}{\ell})$ *Corresponding optimal dual*
REPEAT
  $m \leftarrow m + 1$
  Find weak hypothesis using equation (11) :
  $f_m \leftarrow \mathcal{F}(S, u)$
  Check for optimal solution:
  $If \sum_{i=1}^{\ell} u_i f_i h_m(x_i) \leq \beta, m \leftarrow m - 1, break$
  $H_{im} \leftarrow f_m(x_i)$
  Solve restricted master for new costs:

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

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

6. LPBoost for Regression

The LPBoost algorithm can be extended to optimize any ensemble cost function that can be formulated as a linear program. To solve alternate formulations we need only change the LP restricted master problem solved at each iteration and the criteria given to the base learner. The only assumptions in the current approach are that the number of weak hypotheses be finite and that if an improving weak hypothesis exists then the base learner can generate it. To see a simple example of this consider the problem of boosting regression functions. We use the following adaptation of the SVM regression formulations. This LP was also adapted to boosting using a barrier algorithm in (Rätsch et al., 2000c). We assume we are given a training set of data

$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell))$, but now y_i may take on any real value.

$$\begin{array}{lll} \min_{a,\xi,\xi^*,\epsilon} & \epsilon + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) & \\ s.t. & H_i a - y_i - \xi_i \leq \epsilon, & \xi_i \geq 0,\ i = 1,\ldots,\ell \\ & H_i a - y_i + \xi_i^* \geq -\epsilon, & \xi_i^* \geq 0,\ i = 1,\ldots,\ell \\ & \sum_{i=1}^{m} a_i = 1, & a_i \geq 0,\ i = 1,\ldots,m \end{array} \tag{12}$$

First we reformulate the problem slightly differently:

$$\begin{array}{lll} \min_{a,\xi,\xi^*,\epsilon} & \epsilon + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) & \\ s.t. & -H_i a + \xi_i + \epsilon \geq -y_i, & \xi_i \geq 0,\ i = 1,\ldots,\ell \\ & H_i a + \xi_i^* + \epsilon \geq y_i, & \xi_i^* \geq 0,\ i = 1,\ldots,\ell \\ & -\sum_{i=1}^{m} a_i = -1, & a_i \geq 0,\ i = 1,\ldots,m \end{array} \tag{13}$$

We introduce Lagrangian multipliers (u, u^*, β), construct the dual, and convert to a minimization problem to yield:

$$\begin{array}{ll} \min_{u,u^*,\beta} & \beta + \sum_{i=1}^{\ell} y_i(u_i - u_i^*) \\ s.t. & \sum_{i=1}^{\ell}(-u_i + u_i^*)H_{ij} \leq \beta,\ j = 1,\ldots,m \\ & \sum_{i=1}^{\ell}(u_i + u_i^*) = 1 \\ & 0 \leq u_i \leq C,\quad 0 \leq u_i^* \leq C,\ i = 1,\ldots,\ell \end{array} \tag{14}$$

LP (14) restricted to all weak hypotheses constructed so far becomes the new master problem. If the base learner returns any hypothesis $H_{.j}$ that is not dual feasible, i.e. $\sum_{i=1}^{\ell}(-u_i + u_i^*)H_{ij} > \beta$, then the ensemble is not optimal and the weak hypothesis should be added to the ensemble. To speed convergence we would like the weak hypothesis with maximum deviation, i.e.,

$$max_j \sum_{i=1}^{\ell}(-u_i + u_i^*)H_{ij}. \tag{15}$$

This is perhaps odd at first glance because the criteria do not actually explicitly involve the dependent variables y_i. But within the LPBoost algorithm, the u_i are closely related to the error residuals of the current ensemble. If the data point x_i is overestimated by the current ensemble function by more than ϵ, then by complementarity u_i will be positive and $u_i^* = 0$. So at the next iteration the base learner will attempt to construct a function that has a negative sign at point x_i. If the point x_i falls within the ϵ margin then the $u_i = u_i^* = 0$, and the next base learner will try to construct a function with value 0 at that point. If the data point x_i is underestimated by the current ensemble function by more than ϵ, then by complementarity u_i^* will be positive and $u_i = 0$. So at the next iteration the base learner will attempt to construct a function that has a positive sign at point x_i. By sensitivity analysis, the magnitudes of u and u^* are proportional to the changes of the objective with respect to changes in y.

This becomes even clearer using the approach taken in the Barrier Boosting algorithm for this problem (Rätsch et al., 2000c). Equation (15) can be converted to a least squares problem. For $v_i = -u_i + u_i^*$ and $H_{ij} = f_j(x_i)$,

$$(f(x_i) - v_i)^2 = f(x_i)^2 - 2v_i' f(x_i) + v_i^2. \tag{16}$$

So the objective to be optimized by the base learner can be transformed as follows:

$$\begin{aligned} max_j \sum_{i=1}^{\ell}(-u_i + u_i^*) f_j(x_i) &= min_j \sum_{i=1}^{\ell} v_i f_j(x_i) \\ &= \min_j \frac{1}{2} \sum_{i=1}^{\ell} \left[(f_j(x_i) - v_i)^2 - f_j(x_i)^2 - v_i^2 \right]. \end{aligned} \tag{17}$$

The constant term v_i^2 can be ignored. So effectively the base learner must construct a regularized least squares approximation of the residual function.

The final regression algorithm looks very much like the classification case. The variables u_i and u_i^* can be initialized to any initial feasible point. We present one such strategy here assuming that D is sufficiently large. Here $(a)_+ := \max(a, 0)$ denotes the plus function.

Algorithm 6.1 (LPBoost-Regression).

Given as input training set: S
$m \leftarrow 0$ *No weak hypotheses*
$a \leftarrow 0$ *All coefficients are* 0
$\beta \leftarrow 0$
$u_i \leftarrow \frac{(-y_i)_+}{||y||_1}$ *Corresponding feasible dual*
$u_i^* \leftarrow \frac{(y_i)_+}{||y||_1}$
REPEAT
$m \leftarrow m + 1$
Find weak hypothesis using equation (17) :
$h_m \leftarrow \mathcal{H}(S, (-u + u^*))$
Check for optimal solution:
$If \sum_{i=1}^{\ell}(-u_i + u_i^*)h_m(x_i) \leq \beta, m \leftarrow m - 1, break$
$H_{im} \leftarrow h_m(x_i)$
Solve restricted master for new costs:

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

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

7. Hard Margins, Soft Margins, and Sparsity

The column generation algorithm can also be applied to the hard margin LP error function for boosting. In fact the DualLPBoost proposed by Grove and Schuurmans (Grove & Schuurmans, 1998) does exactly this. Breiman in (Breiman, 1999) also investigated an equivalent formulation using an asymptotic algorithm. Both papers found that optimizing the hard margin LP to construct ensembles did not work well in practice. In contrast the soft margin LP ensemble methods optimized using column generation investigated in this paper and using an arcing approach in (Rätsch et al., 2000b) worked well (see Section 8). Poor performance of hard margin versus soft margin classification methods have been noted in other contexts as well. In a computational study of the hard margin Multisurface-Method (MSM) for classification (Mangasarian, 1965) and the soft margin Robust Linear Programming (RLP) method (Bennett & Mangasarian, 1992) (both closely related LP

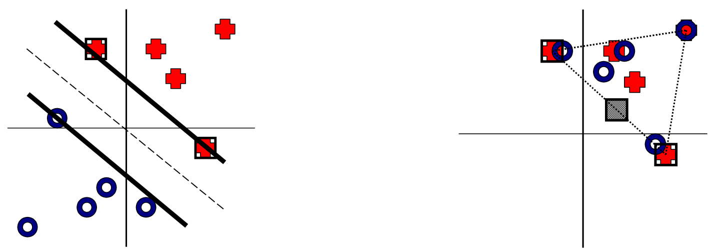


Figure 1. No noise Hard Margin LP solution for two confidence-rated hypotheses. Left is the separation in label space. Right is the separation in dual or margin space.

precursors to Boser et al.'s Support Vector Machine (Boser et al., 1992; Cortes & Vapnik, 1995)), the soft margin RLP performed uniformly better than the hard margin MSM.

In this section we will examine the critical difference between hard and soft margin classifiers geometrically through a simple example. This discussion will also illustrate some of the practical issues of using a column generation approach to solving the soft margin problems.

The hard margin ensemble LP found in (Grove & Schuurmans, 1998) expressed in the notation of this paper is:

$$\begin{array}{ll} \max_{a,\rho} & \rho \\ s.t. & y_i H_i a \geq \rho, i = 1, \ldots, \ell \\ & \sum_{i=1}^{m} a_j = 1, \\ & a_i \geq 0, \; i = 1, \ldots, m \end{array} \tag{18}$$

This is the primal formulation. The dual of the hard margin problem is

$$\begin{array}{ll} \min_{u,\beta} & \beta \\ s.t. & \sum_{i=1}^{\ell} u_i y_i H_{ij} \leq \beta, \; j = 1, \ldots, m \\ & \sum_{i=1}^{\ell} u_i = 1, \quad 0 \leq u_i, \; i = 1, \ldots, \ell \end{array} \tag{19}$$

Let us examine geometrically what the hard and soft margin formulations do using concepts used to described the geometry of SVM in (Bennett & Bredensteiner, 2000). Consider the LP subproblem in the column generation algorithm after sufficient weak hypotheses have been generated such that two classes are linearly separable. Specifically, there exist $\rho > 0$ and a such that $y_i H_i a \geq \rho > 0$ for $i = 1, \ldots, \ell$. Figure (1) gives an example of two confidence rated hypotheses (labels between 0 and 1). The left figure shows the separating hyperplane in the

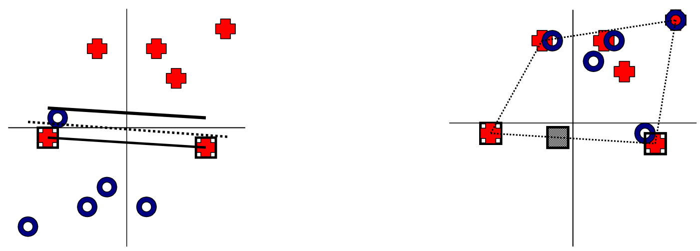


Figure 2. Noisy Hard Margin LP solution for two confidence-rated hypotheses. Left is the separation in label space. Right is separation in dual or margin space.

label space where each data point x_i is plotted as $(h_1(x_i), h_2(x_i))$. The separating hyperplane is shown as a dotted line through the origin as there is no threshold. The minimum margin ρ is positive and produces a very reasonable separating plane. The solution depends only on the two support vectors indicated by boxes. The right side shows the problem in dual or margin space where each point is plotted as $(y_i h_1(x_i), y_i h_2(x_i))$. Recall, a weak hypothesis is correct on a point if $y_i h(x_i)$ is positive. The convex hull of the points in the dual space is shown with dotted lines. The dual LP computes a point in the convex hull[2] that is optimal by some criteria. When the data are linearly separable, the dual problem finds the point $C = \sum_{i=1}^{\ell} u_i y_i H_i$ in the convex hull closest to the origin as measured by the infinity norm. This optimal point is indicated by a shaded square. It is a convex combination of two support vectors which happen to be from the same class. Recall that this dual vector will be used as the misclassification costs for the base learner in the next iteration. In general, there exists an optimal solution for a hard margin LP with k hypotheses with at most k positive support vectors (and frequently much less than k). Note that in practice the hard margin LP may be highly degenerate especially if the number of points is greater than the number of ensembles, so there will be many alternative optimal solutions that are not necessarily as sparse. But a simplex based solver will find the sparse extreme point solutions.

In Figure 2, one noisy point that has been misclassified by both hypotheses has been added. As shown on the left hand side, the optimal separating plane in the label space completely changes. The data in

[2] A point c is in the convex hull of a set of points $q_1, \ldots, q_\ell$ if and only if there exists $u_i \geq 0,\ i = 1, .., \ell$ with $\sum_{i=1}^{\ell} u_i = 1$, such that $c = \sum_{i=1}^{\ell} u_i q_i$.

label space is no longer separable. Consider the dual solution as shown in the margin space on the right hand side. Note that the convex hull in the margin space now intersects the negative orthant. Thus the optimal dual β will be negative and this implies that the margin ρ will also be negative. The dual problem tries to minimize β, thus it will calculate the point in the convex hull intersected with the negative orthant that is furthest from the origin as measured by the infinity norm. This point is indicated by a shaded square. It is determined by two support vectors, the noisy point plus one point in the same class. The single noisy point has had a dramatic effect on the solution. Now almost no weight is being placed on the first hypothesis. Because of the negative margin, one of the circle points is now misclassified but it does not appear as a support vector, i.e., its dual multiplier is 0.

Thus even in this simple example, we can identify some potential issues with using the hard margin LP in an iterative column generation algorithm where the duals are used as misclassification costs such as in (Grove & Schuurmans, 1998). First, the hard margin LP is extremely sensitive to noisy data. Second, the optimal dual vertex solutions are extremely sparse, never exceeding the number of hypotheses considered in the LP. This is a good property when large numbers of hypotheses are considered but can lead to problems in the early stage of the algorithm. Second, the support vectors may be extremely skewed to one class or even all from one class. Third, misclassified points are not necessarily support vectors, thus their misclassification cost will appear as zero to the base learner in the next iteration.

The soft margin formulation addresses some of these problems. To conserve space we will jump straight to the noisy case. Figure 3 illustrates the solutions found by the soft margin LP (see equation (2)) both in the original label space on the left and the dual margin space on the right. On the left side, we see that the separating plane in the hypothesis space is very close to that of the hard margin solution for the no-noise case shown in Figure 1. This seems to be a desirable solution. In general, the soft margin formulation is much less sensitive to noise.

There are some notable differences in the dual solution shown on the right side of Figure 3. The dual LP for the soft margin case is almost identical to the hard margin case with one critical difference: the dual variables are now bounded above. For clarity we repeat the dual LP here.

$$\begin{array}{ll} \min_{u,\beta} & \beta \\ s.t. & \sum_{i=1}^{\ell} u_i y_i H_{ij} \le \beta,\ j = 1, \dots, m \\ & \sum_{i=1}^{\ell} u_i = 1, \quad 0 \le u_i \le D,\ i = 1, \dots, \ell \end{array} \tag{20}$$

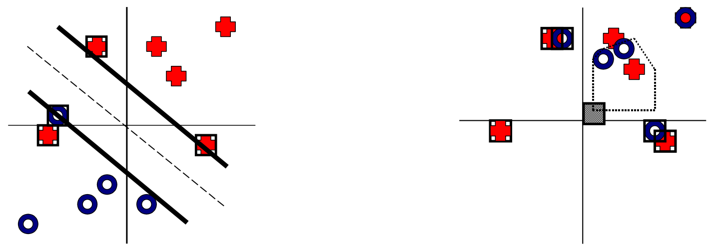


Figure 3. Noisy Soft Margin LP solution for two confidence-rated hypotheses. Left is the separation in label space. Right is the separation in dual or margin space.

In our example, we used a misclassification cost in the primal of $D = 1/4$. In the dual, this has the effect effect of reducing the set of feasible dual points to a smaller set called the reduced convex hull (Bennett & Bredensteiner, 2000). If D is sufficiently small, the reduced convex hull no longer intersects the negative orthant and we once again can return to the case of finding the closest point in the now reduced convex hull to the origin as in the linearly separable case. By adding the upper bound to the dual variables, any optimal dual vector u will still be sparse but not as sparse as in the hard margin case. For $D = 1/k$ it must have at least k positive elements. In the example, there are four such support vectors outlined in Figure 3 with squares. For D sufficiently small, by the LP complementarity conditions , any misclassified point will have a corresponding positive dual multiplier. In this case the support vectors also were drawn from both classes, but note that there is nothing in the formulation that guarantees this.

To summarize, if we are calculating the optimal hard margin ensemble over a large enough hypothesis space such that the data is separable in the label space, it may work very well. But in the early iterations of a column generation algorithm, the hard margin LP will be optimized over a small set of hypotheses such that the classes are not linearly separable in the label space. In this case we observe several problematic characteristics of the hard margin formulation: extreme sensitivity to noise (producing undesirable hypotheses weightings), extreme sparsity of the dual vector especially in the early iterations of a column generation algorithm, failure to assign positive Lagrangian multipliers to misclassified examples, and no guarantee that points will be drawn from both distributions. Although we examined these potential problems using the confidence-rated case in 2 dimensions it is easy to see that they hold true and are somewhat worse for the more typical case where the labels are restricted to 1 and -1.

The soft margin LP adopted in this paper addresses some but not all of these problems. Adding soft margins makes the LP much less sensitive to noise. Adding soft margins to the primal corresponds to adding bounds on the dual multipliers. The constraint that the dual multipliers sum to one forces more of the multipliers to be positive both in the separable and inseparable cases. Furthermore the complementarity conditions of the soft margin LP guarantee that any point that violates the soft margin will have a positive multiplier. Assuming D is sufficiently small, this means that every misclassified point will have a positive multiplier.

But this geometric analysis illustrates that there are some potential problems with the soft margin LP. The column generation algorithm uses the dual costs as misclassification costs for the base learner to generate new hypotheses. So the characteristics of the dual solution are critical. For a small set of hypotheses, the LP will be degenerate, and the dual solution may still be quite sparse. Any method that finds extreme point solutions will be biased to the sparsest dual optimal solution, when in practice less sparse solutions would be better suited as misclassification costs for the base learner. If the parameter D is chosen too large the margin may still be negative so the LP will still suffer from the many problems found in the hard margin case. If the parameter D is chosen too small then the problem reduces to the equal cost case so little advantage will be gained through using an ensemble method. Potentially, the distribution of the support vectors may still be highly skewed towards one class. All of these are potential problems in an LP-Based ensemble method. As we will see in the following sections, they can arise in practice.

8. Computational Experiments

We performed three sets of experiments to compare the performance of LPBoost, CRB, and AdaBoost on three classification tasks: one boosting decision tree stumps on smaller datasets and two boosting C4.5 (Quinlan, 1996). For decision tree stumps six datasets were used, LP-Boost was run until the optimal ensemble was found, and AdaBoost was stopped at 100 and 1000 iterations . For the C4.5 experiments, we report results for four large datasets with and without noise. Finally, to further validate C4.5, we experimented with ten more additional datasets. The rationale was to first evaluate LPBoost where the base learner solves (10) exactly and the optimal ensemble can be found by LP-Boost. Then our goal was to examine LPBoost in a more realistic environment by using C4.5 as a base learner using a relatively

Table I. Average Accuracy and Standard Deviations of Boosting using Decision Tree Stumps; (m) = average number of unique decision tree stumps in final ensemble

Dataset	LPBoost (m)	AB-100(m)	AB-1000(m)
Cancer	0.966 ± 0.025 (14.7)	0.954 ± 0.029 (36.8)	0.947 ± 0.026 (59.3)
Diagnostic	0.961 ± 0.027 (54.2)	0.968 ± 0.027 (67.7)	0.970 ± 0.031 (196.1)
Heart	0.795 ± 0.079 (70.8)	0.818 ± 0.075 (51.1)	0.801 ± 0.061 (103.1)
Ionosphere	0.906 ± 0.052 (87.6)	0.906 ± 0.054 (69.1)	0.903 ± 0.043 (184.2)
Musk	0.882 ± 0.035 (205.3)	0.840 ± 0.042 (89.8)	0.891 ± 0.033 (370.9)
Sonar	0.870 ± 0.082 (85.7)	0.808 ± 0.084 (76.4)	0.856 ± 0.078 (235.5)

small number of maximum iterations for both LPBoost and AdaBoost. All of the datasets were obtained from the UC-Irvine data repository (Murphy & Aha, 1992). For the C4.5 experiments we performed both traditional and confidence-rated boosting. Different strategies for picking the LP model parameter were used in each of the three type to make sure the results were not a quirk of any particular model selection strategy. The implementations of LPBoost were identical except in how the misclassification costs were generated and in the stopping criteria. Both methods were allowed the same maximum number of iterations.

8.1. Boosting Decision Tree Stumps

We used decision tree stumps as base hypotheses on the following six datasets: Cancer (9,699), Diagnostic (30,569), Heart (13,297), Ionosphere (34,351), Musk (166,476), and Sonar (60,208). The number of features and number of points in each dataset are shown, respectively, in parentheses. We report testing set accuracy for each dataset based on 10-fold Cross Validation (CV). We generate the decision tree stumps based on the mid-point between two consecutive values for a given variable. Since there is limited confidence information in stumps, we did not perform confidence-rated boosting. All boosting methods search for the best weak hypothesis which returns the least weighted misclassification error at each iteration. LPBoost can take advantage of the fact that each weak hypothesis need only be added into the ensemble once. Thus once a stump is added to the ensemble it is never evaluated by the learning algorithm again. The weights of the weak hypotheses are adjusted dynamically by the LP. This is an advantage over AdaBoost, since AdaBoost adjust weights by repeatedly adding the same weak hypothesis into the ensemble. As is discussed in Section 8.3, the computational effort to reoptimize the LP is a fraction of the time to find a weak hypothesis.

The parameter ν for LPBoost was set using a simple heuristic: 0.1 added to previously-reported linear discriminant 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, 0.1, 0.25, 0.2, 0.25, 0.3). The parameter ν corresponds to the fraction of the data that are support vectors which we overestimate as the error estimate plus 10 percent. The base linear error estimate can easily be derived using cross-validation instead of using results from a previous paper. This is rough heuristic that gives a reasonable guess at ν but further tuning may improve it. For cancer the initial guess of $\nu = 0.15$ overfit the training data so it was increased to $\nu = 0.2$. Results for AdaBoost were reported for a maximum number of iterations of 100 and 1000. Many authors have reported results for AdaBoost at these iterations using decision stumps. The 10-fold average classification accuracies and standard deviations are reported in Table I. We also report the average number of unique weak hypotheses over 10 folds.

LPBoost performed well in terms of classification accuracy, number of weak hypotheses, 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. 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 criterion for stopping when an optimal ensemble is found that is reached in relatively few iterations. It uses few weak hypotheses. There are only 81 possible stumps on the Breast Cancer dataset (nine attributes having nine possible values), so clearly AdaBoost may require the same tree to be generated multiple times. LPBoost generates a weak hypothesis only once and can alter the weight on that weak hypothesis at any iteration. The run time of LPBoost is proportional to the number of weak hypotheses generated. Since the LP package that we used, CPLEX 4.0 (CPL, 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 that an iteration of AdaBoost. The time is proportional to the number of weak hypotheses generated. For problems in which LPBoost generates far fewer weak hypotheses it is much less computationally costly. Results also clearly indicate that if AdaBoost uses fewer unique weak hypotheses, it underfits. In the opposite case, it overfits. LPBoost depends on the choice of the model parameter for preventing overfitting. AdaBoost depends on choice of the maximum number of iterations to prevent overfitting.

In the next subsection, we test the practicality of our methodology on different datasets using C4.5 in a more realistic environment where both AdaBoost and LPBoost are halted after a relatively small number of iterations.

8.2. Boosting C4.5

LPBoost with C4.5 as the base algorithm performed well after some operational challenges were solved. In concept, boosting using C4.5 is straightforward since the C4.5 algorithm accepts misclassification costs. One problem is that C4.5 only finds a good solution not guaranteed to maximize (10). This can effect the convergence speed of the algorithm and may cause the algorithm to terminate at a suboptimal solution. As discussed in Section 7 another challenge is that the misclassification costs determined by LPBoost are very 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. In practice we found that when many $u_i = 0$, LPBoost converged slowly. In the limited number of iterations that we allowed (e.g. 25), LPBoost frequently failed to find weak hypotheses that improved significantly over the initial equal cost solution. The weak hypotheses generated using only subsets of the variables were not necessarily good over the full data set. Thus the search was too slow. Alternative optimization algorithms may alleviate this problem. For example, an interior point strategy may lead to significant performance improvements. When LPBoost was solved to optimality on decision tree stumps with full evaluation of the weak hypotheses, this problem did not occur. Boosting unpruned decision trees helped somewhat but did not completely eliminate this problem.

Stability and convergence speed were greatly improved by adding minimum misclassification costs to the dual LP (5) :

$$\begin{array}{ll} \min_u & \beta \\ s.t. & \sum_{i=1}^{\ell} u_i y_i H_{ij} \leq \beta,\ j = 1,\dots,m \\ & \sum_{i=1}^{\ell} u_i = 1 \\ & D' \leq u_i \leq D,\ i = 1,\dots,\ell \end{array} \tag{21}$$

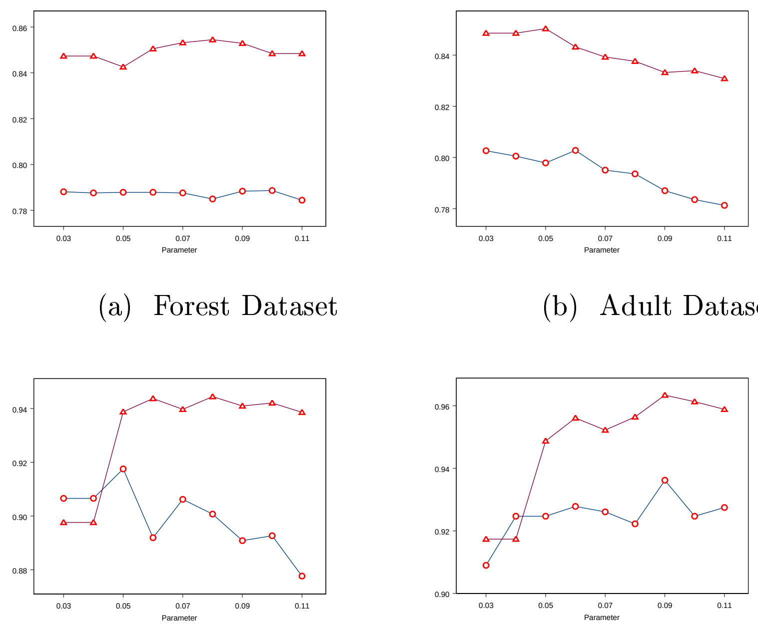
where $D = \frac{1}{\nu\ell}$ and $D' = \frac{1}{25\nu\ell}$. The corresponding primal problem is

$$
\begin{array}{ll}
\max_{a,\xi,\rho} & \rho + D' \sum_{i=1}^{\ell} \tau_i - D \sum_{i=1}^{\ell} \xi_i \\
s.t. & y_i H_i a + \xi_i \geq \rho + \tau_i, \ i = 1, \dots, \ell \\
 & \sum_{i=1}^{m} a_i = 1, \ , a_i \geq 0, \ i = 1, \dots, m \\
 & \xi_i \geq 0, \ i = 1, \dots, \ell
\end{array}
\tag{22}
$$

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.

LPBoost was adapted to the multiclass problem using a similar approach to that used in (Grove & Schuurmans, 1998). Specifically, $h_j(x_i) = 1$ if instance x_i is correctly classified in the appropriate class by weak hypothesis h_j and -1 otherwise. Similarly, the same approach has been used to apply AdaBoost to multiclass problems. AdaBoost increases the weight of a certain point if it is misclassified and decreases the weight otherwise. For both multiclass LPBoost and AdaBoost, the predicted class is the class which achieves a majority in a vote weighted by the ensemble weights of the final set of weak hypotheses. This is just a very simple method of boosting multiclass problems. Further investigation of LP multiclass approaches is needed. We ran experiments on larger datasets: Forest, Adult, USPS, and Optdigits from UCI(Murphy & Aha, 1992). Forest is a 54-dimension dataset with seven possible classes. The data are divided into 11340 training, 3780 validation, and 565892 testing instances. There are no missing values. The 15-dimensional Adult dataset has 32562 training and 16283 testing instances. One training point that 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. USPS and Optdigits are optical character recognition datasets. USPS has 256 dimensions without missing values. Out of 7291 original training points, we use 1822 points as training data and the other 5469 as validation data. There are 2007 test points. Optdigits on the other hand has 64 dimensions without missing values. Its original training set has 3823 points. We use 955 of them as training data and the remaining 2868 as validation data. A stratified sampling strategy is used to preserve the original distributions of training sets after partitioning.

As suggested in (Bauer & Kohavi, 1999), training sets are selected to be relatively small in order to favor boosting methods. There will be more variance in the weak hypotheses for smaller datasets than for large datasets. The selections of the maximum the number of iterations



(a) Forest Dataset

(b) Adult Dataset

(c) USPS Dataset

(d) Optdigits Dataset

Figure 4. Validation Set Accuracy by ν Value. Triangles are no noise and circles are with noise.

in AdaBoost were based on validation set results. So to give AdaBoost every advantage we used large validation sets. The ν parameter was also chosen using this same validation set. Since initial experiments (not reported here) resulted in the same parameter set for both LPBoost and CRB, we set the parameters equal for CRB and LPBoost to shorten computational work for the validation process. In addition, it is sometimes desirable to report computational results from similar methods using the same parameter sets. In order to investigate the performance of boosted C4.5 with noisy data, we introduced 15% label noise for all four datasets. To avoid problems with underflow of boosting, AdaBoost is modified by removing points with weight values less than 10^{-10} from the learning process. Confidence rates returned from C4.5 at termination nodes are used in CRB. We use positive confidence if a certain point is classified correctly and negative confidence otherwise.

The ν parameter used in LPBoost and the number of iterations of AdaBoost can significantly affect their performance. Thus accuracy on the validation set was used to pick the parameter ν for LPBoost and the number of iterations for AdaBoost. We limit the maximum number

Table II. Large dataset results from boosting C4.5 by method and maximum number of iterations

Method	Iteration	Forest	+15% Noise	Adult	+15% Noise	USPS	+15% Noise	OptDigits	+15% Noise
LPBoost	25	0.7226	0.6602	0.8476	0.8032	0.9123	0.8744	0.9249	0.8948
	50	0.7300	0.6645	0.8495	0.8176	0.9188	0.8849	0.9416	0.9060
	100	0.7322	0.6822	0.8501	0.8461	0.9153	0.9103	0.9449	0.9160
CRB	25	0.7259	0.6569	0.8461	0.8219	0.9103	0.8739	0.9355	0.8948
	50	0.7303	0.6928	0.8496	0.8240	0.9063	0.8789	0.9349	0.9093
	100	0.7326	0.7045	0.8508	0.8250	0.9133	0.8874	0.9343	0.9238
AdaBoost	25	0.7370	0.6763	0.8358	0.7630	0.9130	0.8789	0.9416	0.8770
	50	0.7432	0.6844	0.8402	0.7630	0.9188	0.8934	0.9494	0.9104
	100	0.7475	0.6844	0.8412	0.7752	0.9218	0.8889	0.9510	0.9243
C4.5	1	0.6638	0.5927	0.8289	0.7630	0.7833	0.6846	0.7958	0.6884

Table III. Stopping iterations determined by validation for AdaBoost

	25 iterations		50 iterations		100 iterations	
Dataset	Original	Noisy	Original	Noisy	Original	Noisy
Forest	22	19	36	39	51	39
Adult	25	4	48	4	74	91
USPS	22	25	47	40	86	99
Optdigits	25	25	49	50	91	94

of iterations at 25, 50 and 100 for all boosting methods. We varied parameter ν between 0.03 and 0.11. Initial experiments indicated that for very small ν values, LPBoost results in one classifier which assigns all training points to one class. On the other extreme, for larger values of ν, LPBoost returns one classifier which is equal to the one found in the first iteration. Figure 4 shows the validation set accuracy for LPBoost on all four datasets with the maximum number of iterations at 25. LPBoost and CRB were not tuned again for 50 and 100 maximum number of iterations. The best validation set accuracy was used for early stopping of AdaBoost in all experiments. So in some sense, AdaBoost has the advantage in these experiments.

Based on validation set results at 25, 50 and 100 the maximum number of iterations, we find the best AdaBoost validation set results at the number of iterations reported in Table III for both original and 15% noisy data. The testing set results using the value of ν with the best validation set accuracy as found for the 25 iteration case are given in Table II. As seen in Table II, LPBoost is comparable with AdaBoost in terms of classification accuracy when the validation set is used to pick the best parameter settings. All boosting methods perform equally well. Although there is no parameter tuning for CRB, it performs very well on the noisy data. All boosting methods outperform C4.5. Results also indicate that none of the boosting methods overfits badly. This can be explained by early stopping based on large validation sets.

We also conducted experiments by boosting C4.5 on small datasets. Once again there was no strong evidence of superiority of any of the boosting approaches. In addition to six UCI datasets used in decision tree stumps experiments, we use four additional UCI datasets here. These are the House(16,435), Housing(13,506)[3], Pima(8,768), and Spam(57,4601) datasets. As in the decision tree stumps experiments, we report results from 10-fold CV. Since the best ν value for LPBoost varies between 0.03 and 0.11 for the large datasets, we pick parameter $\nu = 0.07$ for the small datasets. No effort was made to tune the ν parameter. Thus there is no advantage given to LPBoost. All boosting methods were allowed to run up to 25 iterations. Results are reported in Table IV. C4.5 performed the best on the House dataset. AdaBoost performed the best in four datasets out of ten. LPBoost and CRB had the best classification performance for three and two datasets respectively. When we drop CRB in Table IV, LPBoost would in this case perform the best in five datasets.

8.3. Computational Cost Analysis

In this section, we analyze computational costs of different boosting methods. One important issue is to justify the additional cost of LP time in LPBoost and CRB. Is it worth reoptimizing an LP at each iteration? Since we use a column generation approach, in theory, it should not affect performance very much. In order to report timings, we reran some of the experiments on a fully dedicated IBM RS-6000 with 512MB RAM and a single 330MhZ processor. Results were consistent across different datasets so we focus on two sample cases. We plot CPU time in seconds per iteration for a large dataset (single run

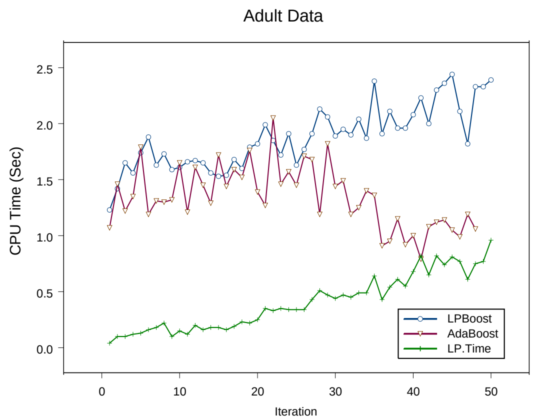
[3] The continuous response variable of the Housing dataset was categorized at 21.5.

Table IV. Small Dataset Results from Boosting C4.5

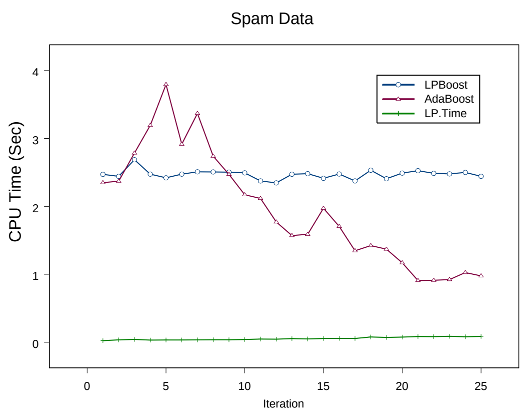
Dataset	LPBoost	CRB	AdaBoost	C4.5
Cancer	0.959 ± 0.017	0.963 ± 0.025	0.966 ± 0.025	0.945 ± 0.025
Diagnostic	0.965 ± 0.026	0.963 ± 0.028	0.971 ± 0.019	0.937 ± 0.037
Heart	0.791 ± 0.062	0.795 ± 0.010	0.787 ± 0.061	0.788 ± 0.077
House	0.959 ± 0.034	0.945 ± 0.053	0.951 ± 0.042	0.962 ± 0.029
Housing	0.854 ± 0.048	0.866 ± 0.038	0.879 ± 0.039	0.817 ± 0.049
Ionosphere	0.937 ± 0.038	0.926 ± 0.060	0.936 ± 0.041	0.916 ± 0.052
Musk	0.882 ± 0.054	0.906 ± 0.049	0.929 ± 0.028	0.834 ± 0.034
Pima	0.750 ± 0.050	0.728 ± 0.048	0.748 ± 0.071	0.729 ± 0.046
Sonar	0.817 ± 0.083	0.832 ± 0.083	0.814 ± 0.093	0.701 ± 0.073
Spam	0.956 ± 0.009	0.955 ± 0.010	0.952 ± 0.009	0.930 ± 0.009

for Adult) and one from a relatively smaller dataset (Spam averaged over 10 folds) in Figure 5. The total CPU times for each iteration of AdaBoost and LPBoost are shown. This includes both the time to find a weak hypothesis and the time to determine the ensemble weights at each iteration. In addition we show the subset of total CPU time in each iteration required to reoptimize LPBoost(LP Time).

Figure 5 clearly demonstrates that LPBoost is computationally tractable. For both the small and large datasets, the computational costs of the base learner, C4.5, far outweighs the cost of updating the LP at each iteration. For the small dataset, the CPU time per iteration is roughly constant. For the large dataset, we can see linear growth in the LP time per iteration. but the iterations are only taking a couple seconds each. In general the time for an LPBoost iteration and AdaBoost per iteration are on the same order of magnitude since the weak learner cost dominates. But one can also see that as the number of iterations increases, the computational costs of AdaBoost are actually decreasing. This is because points with small weights are left out of the dataset for AdaBoost. This is because the weights on individual points have become sufficiently small such that they may be dropped from the training data. Recall that to prevent excessive sparsity in the early iterations of LPBoost we introduced a lower bound on the weights. This improved the performance of LPBoost in the early iterations but at computational penalty in later iterations. AdaBoost can be regarded as a barrier method (Rätsch et al., 2000c) so it achieves sparsity of the misclassification costs only in the limit. LPBoost finds extreme point solutions so without the lower bound the early LP iterations would actually be more sparse in the beginning. Our strategy of adding a lower bound makes sense in the early iterations, but as the set of weak hypotheses grows it could be dropped in order to have the same performance behavior as AdaBoost. The best LP formulation for boosting is



(a) Adult Dataset



(b) Spam Dataset Average 10 fold CV

Figure 5. CPU times in seconds for each iteration

still very much an open question. But it is clear that column generation is a very practical, tractable approach to boosting with computational costs per iteration similar to AdaBoost. To further support this, we also report total run times for LPBoost and AdaBoost in Table V. For 25 and 50 iterations of boosting C4.5, little difference is seen between the two approaches.

Table V. Representative total run times for large datasets

	25 iterations		50 iterations	
Dataset	LPBoost	AdaBoost	LPBoost	AdaBoost
Forest	349	326	770	604
Adult	39	34	94	64
USPS	95	138	196	272
Optdigits	11	12	24	25

9. 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. The LPBoost algorithm can be applied to any boosting problem formulated as an LP. We examined algorithms based on the 1-norm soft margin cost functions for support vector machines. A generalization error bound was found for the classification case. The LP optimality conditions allowed us to provide explanations for how the methods work. In classification, the dual variables act as misclassification costs. The optimal ensemble consists of a linear combination of weak hypotheses that work best under the worst possible choice of misclassification costs. This explanation is closely related to that of (Breiman, 1999). For regression as discussed in the Barrier Boosting approach to the same formulation (Rätsch et al., 2000c), the dual multipliers act like error residuals to be used in a regularized least square problem. We demonstrated the ease of adaptation to other boosting problems by examining the confidence-rated and regression cases. The hard margin LP algorithm of (Grove & Schuurmans, 1998) is a special case of this general approach. Extensive computational experiments found that the method performed well versus AdaBoost both with respect to classification quality and solution time. We found little clear benefit for confidence-rated boosting of C4.5 decision trees. From an optimization perspective, LPBoost has many benefits over gradient-based approaches: finite termination at a globally optimal solution, well-defined convergence criteria based on optimality conditions, fast algorithms in practice, and fewer weak hypotheses in the optimal ensemble. LPBoost may be more sensitive to inexactness of the base learning algorithm and problems can arise due to the extreme sparsity of the misclassification costs in the early iteration. But through modification of the base LP, we were able to obtain very good performance over a wide spectrum of datasets even when boosting decision trees where the assumptions of the learning algorithm were violated. The questions of what is the best LP formulation for boosting and the best method for optimizing the LP remain open. Interior point column generation algorithms may be much more efficient. But clearly LP formulations for classification and regression 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 Commission 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. In Schölkopf, B., Burges, C., & Smola, A. (Eds.), *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. In M. Kearns, S. Solla, D. C. (Ed.), *Advances in Neural Information Processing Systems 11*, pp. 368–374 Cambridge, MA. MIT Press.

Bennett, K. P., Demiriz, A., & Shawe-Taylor, J. (2000). A column generation approach to boosting. In *Proceedings of International Conference on Machine Learning* Stanford, California. To appear.

Bennett, K. P., & Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, *1*, 23–34.

Bennett, K., & Bredensteiner, E. J. (2000). Duality and geometry in svm classifiers. In Langley, P. (Ed.), *Proceedings of the 17th International Conference on Machine Learning*, pp. 57–64. Morgan Kaufmann.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In Haussler, D. (Ed.), *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152. ACM Press.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, *11*(7), 1493–1517.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273–297.

CPLEX Optimization Incorporated, Incline Village, Nevada (1994). *Using the CPLEX Callable Library.*

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines.* Cambridge University Press.

Grove, A., & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98.*

Mangasarian, O. L. (1965). Linear and nonlinear separation of patterns by linear programming. *Operations Research*, *13*, 444–452.

Mangasarian, O. L. (2000). Generalized support vector machines. In Smola, A., Bartlett, P., Schölkopf, B., & Schuurmans, D. (Eds.), *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.* McGraw-Hill, New York, NY.

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

Rätsch, G., Schölkopf, B., Smola, A., Mika, S., Onoda, T., & Müller, K.-R. (2000a). Robust ensemble learning. In Smola, A., Bartlett, P., Schölkopf, B., & Schuurmans, D. (Eds.), *Advances in Large Margin Classifiers*, pp. 207–219 Cambridge, MA. MIT Press.

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

Rätsch, G., Warmuth, M., Mika, S., Onoda, T., Lemm, S., & Müller, K.-R. (2000c). Barrier boosting. Tech. rep..

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*(5), 1651–1686.

Schapire, R., & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Conference on Computational Learning Theory, COLT'98*, pp. 80–91. to appear in Machine Learning.

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*(5), 1926–1940.

Shawe-Taylor, J., & Cristianini, N. (1999). Margin distribution bounds on generalization. In *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. Tech. rep. RC-21572, IBM.