Totally Corrective Boosting Algorithms that Maximize the Margin

Manfred K. Warmuth MANFRED@CSE.UCSC.EDU
Jun Liao LIAOJUN@CSE.UCSC.EDU
University of California at Santa Cruz, Santa Cruz, CA 95064, USA

Gunnar Rätsch GUNNAR.RAETSCH@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory of the Max Planck Society, Spemannstr. 39, 72076 Tübingen, Germany

Boosting, Margins, Convergence, Relative Entropy, Bregman Divergences, Bregman Projection

Abstract

We consider boosting algorithms that maintain a distribution over a set of examples. At each iteration a weak hypothesis is received and the distribution is updated. We motivate these updates as minimizing the relative entropy subject to linear constraints. For example AdaBoost constrains the *edge* of the last hypothesis w.r.t. the updated distribution to be at most $\gamma = 0$. In some sense, AdaBoost is "corrective" w.r.t. the last hypothesis. A *cleaner* boosting method is to be "totally corrective": the edges of *all* past hypotheses are constrained to be at most γ, where γ is suitably adapted.

Using new techniques, we prove the same iteration bounds for the totally corrective algorithms as for their corrective versions. Moreover with adaptive γ, the algorithms provably maximizes the margin. Experimentally, the totally corrective versions return smaller convex combinations of weak hypotheses than the corrective ones and are competitive with LPBoost, a totally corrective boosting algorithm with no regularization, for which there is no iteration bound known.

*The first author was partially funded by the NSF grant CCR-9821087. The first two authors were partially funded by UC Discovery grant ITl03-10127 and Telik Inc. grant ITl03-10110. Part of this work was done while the third author was visiting UC Santa Cruz. The authors would thank Telik Inc. for providing the COX-1 dataset.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

1. Introduction

In this paper we characterize boosting algorithms by the underlying optimization problems rather than the approximation algorithms that solve these problems. The goal is to select a small convex combination of weak hypotheses that maximize the margin. For lack of space we only compare the algorithms in terms of this goal rather than the generalization error and refer to (Schapire et al., 1998) for generalization bounds that improve with the margin and degrade with the size of the final convex combination.

One of the most common boosting algorithms is AdaBoost (Freund & Schapire, 1997; Schapire & Singer, 1999). It can be viewed as minimizing the relative entropy to the last distribution subject to the constraint that the edge of the last hypothesis is zero (equivalently its weighted error is half) (Kivinen & Warmuth, 1999; Lafferty, 1999). One of the important properties of AdaBoost is that it has a decent iteration bound and approximately maximizes the margin of the examples (Breiman, 1997; Rätsch et al., 2001; Rudin et al., 2004a). A similar algorithm called AdaBoost$^*_\nu$ provably maximizes the margin and has an analogous iteration bound (Rätsch & Warmuth, 2005).[1] This algorithm enforces only a single constraint at iteration t: the edge of the hypothesis must be at most γ, where γ is adapted.

A natural idea is to constrain the edges of *all* t past hypotheses to be at most γ and otherwise minimize the relative entropy to the initial distribution. Such algorithms were proposed by Kivinen and Warmuth (1999) and are called "totally corrective". However, in that paper only $\gamma = 0$ was considered, which leads to

[1]Other algorithms for maximizing the margin with weaker iteration bounds are given in (Breiman, 1999; Rudin et al., 2004a).

an infeasible optimization problem when the training data is separable. Building on the work of Rätsch and Warmuth (2005), we now adapt the edge bound γ of the totally corrective algorithm so that the margin is approximately maximized. We call our new algorithm TotalBoost$_\nu$.

The corrective AdaBoost$^*_\nu$ can be used as a heuristic for implementing TotalBoost$_\nu$ by doing many passes over all past hypotheses before adding a new one. However, we can show that this heuristic is often several orders of magnitude less efficient than a vanilla *sequential quadratic optimization* approach for solving the optimization problem underlying TotalBoost$_\nu$.

A parallel progression occurred for on-line learning algorithms of disjunctions. The original algorithms (variants of the Winnow algorithm (Littlestone, 1988)) can be seen as processing a single constraint induced by the last example. However, more recently an on-line algorithm has been developed for learning disjunctions (in the noise-free case) that enforces the constraints induced by all past examples (Long & Wu, 2005). The proof techniques in both settings are essentially the same except that for disjunctions the margin/threshold is fixed whereas in boosting we optimize the margin.

Besides emphasizing the new proof methods for iteration bounds of boosting algorithms, this paper also does an experimental comparison of the algorithms. We show that while TotalBoost$_\nu$ has the same iteration bound as AdaBoost$^*_\nu$, it often requires several orders of magnitudes fewer iterations. When there are many similar weak hypotheses, the totally corrective algorithms has an additional advantage: assume we have 100 groups of 100 weak hypotheses each, where the hypotheses within each group are very similar. TotalBoost$_\nu$ picks a small number of hypotheses from each group, whereas the algorithms that process one constraint at a time often come back to the same group and choose many more members from the same group. Therefore in our experiments the number of weak hypotheses in the final convex combination (with non-zero coefficients) is consistently much smaller for the totally corrective algorithms, making them better suited for the purpose of feature selection.

Perhaps one of the simplest boosting algorithms is LPBoost: it is totally corrective, but unlike TotalBoost$_\nu$, it uses no entropic regularization. Also, the upper bound γ on the edge is chosen to be as small as possible in each iteration, whereas in TotalBoost$_\nu$ it is decreased more moderately. Experimentally, we have identified cases where TotalBoost$_\nu$ requires considerably fewer iterations than LPBoost, which suggests that either the entropic regularization or the moderate choice of γ is helpful for more than just for proving iteration bounds.

2. Preliminaries

Assume we are given N labeled examples $(\mathbf{x}_n, y_n)_{1\leq n\leq N}$, where the examples are from some domain and the labels y_n lie in $\{\pm 1\}$. A boosting algorithm combines many "weak" hypotheses or rules of thumb for the examples to form a convex combination of hypotheses with high accuracy. In this paper a boosting algorithm adheres to the following protocol: it maintains a distribution $\mathbf{d}^t$ on the examples; in each iteration t a weak learner provides a "weak" hypothesis h^t and the distribution $\mathbf{d}^t$ is updated to $\mathbf{d}^{t+1}$. Intuitively the updated distribution incorporates the information obtained from h^t and gives high weights to the remaining "hard" examples. After iterating T steps the algorithm stops and outputs a convex combination of the T weak hypotheses it received from the weak learner.

We first discuss how we measure the performance of a weak hypothesis h w.r.t. the current distribution $\mathbf{d}$. If h is ± 1 valued, then the error ϵ is the total weight on all the examples that are misclassified. When the range of a hypothesis h is the entire interval $[-1, +1]$, then the *edge* $\gamma_h(\mathbf{d}) = \sum_{n=1}^N d_n y_n h(\mathbf{x}_n)$ is a more convenient quantity for measuring the quality of h. This edge is an affine transformation of the error for the case when h has range ± 1: $\epsilon_h(\mathbf{d}) = \frac{1}{2} - \frac{1}{2}\gamma_h(\mathbf{d})$. Ideally we want a hypothesis of edge 1 (error 0). On the other hand it is often easy to produce hypotheses of edge at least 0 (or equivalently error at most $\frac{1}{2}$). We define the edge of a set of hypotheses as the maximum of the edges.

Assumption on the weak learner: Assume that for any distribution $\mathbf{d}$ on the examples the weak learner returns a hypothesis h with edge $\gamma_h(\mathbf{d})$ at least g. As we will discuss later, the *guarantee* parameter g might not be known to the boosting algorithm.

Boosting algorithms produce a convex combination of weak hypotheses: $f_{\boldsymbol{\alpha}}(\mathbf{x}) := \sum_{t=1}^T \alpha_t h^t(\mathbf{x})$, where h^t is the hypothesis added in iteration t and α_t is its coefficient. The *margin* of a given example $(\mathbf{x}_n, y_n)$ is defined as $y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)$. The margin of a set of examples is always the minimum over the examples.

Our algorithms always produce a convex combination of weak learners of margin at least $g - \nu$, where ν is a precision parameter. Also the size of the convex combination is at most $O(\frac{\log N}{\nu^2})$. Note that the higher the guarantee g of the weak learner, the larger the produced margin.

Algorithm 1 LPBoost algorithm

```
1. Input: $S = \langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\rangle$, desired accuracy $\nu$
2. Initialize: $d^1_n = 1/N$ for all $n = 1 \ldots N$
3. Do for $t = 1, \ldots,$
   (a) Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h^t : \mathbf{x} \mapsto [-1, 1]$ and let $u^t_n = y_n h^t(\mathbf{x}_n)$
   (b) Calculate the edge $\gamma_t$ of $h^t$: $\gamma_t = \mathbf{d}^t \cdot \mathbf{u}^t$,
   (c) Set $\widehat{\gamma}_t = (\min_{q=1,\ldots,t} \gamma_q) - \nu$
   (d) Compute $\gamma^*_t$ as in (1) and set $\mathbf{d}^{t+1}$ to any distribution $\mathbf{d}$ for which $\mathbf{u}^q \cdot \mathbf{d} \le \gamma^*_t$, for $1 \le q \le t$
   (e) If $\gamma^*_t \ge \widehat{\gamma}_t$ then $T = t$ and break
4. Output: $f_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{t=1}^T \alpha_t h^t(\mathbf{x})$, where the coefficients $\alpha_t$ realize margin $\gamma^*_T$.
```

How are edges and margins related? By duality the minimum edge of the examples w.r.t. the hypotheses set $\mathcal{H}^t = \{h^1, \ldots, h^t\}$ equals the maximum margin:

$$\gamma^*_t := \min_{\mathbf{d}} \max_{h \in \mathcal{H}^t} \gamma_h(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \min_n y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) := \rho^*_t, \quad (1)$$

where $\mathbf{d}$ and $\boldsymbol{\alpha}$ are N and t dimensional probability vectors, respectively. Note that the sequence γ^*_t is non-decreasing. It will approach the guarantee g from below. The algorithms will stop as soon as the edges are within ν of g (See next section.)

The above duality also restricts the range of the guarantee g that a weak learner can possible have. Let $\mathcal{H}$ be the entire (possibly infinite) hypothesis set from which the weak learner is choosing. If $\mathcal{H}$ is compact (see discussion in Rätsch & Warmuth, 2005) then

$$\gamma^* := \min_{\mathbf{d}} \max_{h \in \mathcal{H}} \gamma_h(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \min_n y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) := \rho^*,$$

where $\mathbf{d}$ and $\boldsymbol{\alpha}$ are probability distributions over the examples and $\mathcal{H}$, respectively, and $f_{\boldsymbol{\alpha}}(\mathbf{x}_n)$ now sums over $\mathcal{H}$. Clearly $g \le \rho^*$ and for any non-optimal $\mathbf{d}, \boldsymbol{\alpha}$:

$$\max_{h \in \mathcal{H}} \gamma_h(\mathbf{d}) > \gamma^* = \rho^* > \min_n y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) =: \rho(\boldsymbol{\alpha}). \quad (2)$$

So even though there always is a weak hypothesis in $\mathcal{H}$ with edge at least ρ^*, the weak learner is only guaranteed to produce one of edge at least $g \le \rho^*$.

One of the most bare-bones boosting algorithms is LPBoost (Algorithm 1) proposed by Grove and Schuurmans (1998); Bennett et al. (2000). It uses linear programming to constrain the edges of the past t weak hypotheses to be at most γ^*_t, which is as small as possible. No iteration bound is known for this algorithm,

Algorithm 2 TotalBoost$_\nu$ with accuracy param. ν

```
1. Input: $S = \langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\rangle$ , desired accuracy $\nu$
2. Initialize: $d^1_n = 1/N$ for all $n = 1 \ldots N$
3. Do for $t = 1, \ldots$
   (a) Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h^t : \mathbf{x} \mapsto [-1, 1]$ and let $u^t_n = y_n h^t(\mathbf{x}_n)$
   (b) Calculate the edge $\gamma_t$ of $h^t$: $\gamma_t = \mathbf{d}^t \cdot \mathbf{u}^t$
   (c) Set $\widehat{\gamma}_t = (\min_{q=1,\ldots,t} \gamma_q) - \nu$
   (d) Update weights:
       $\mathbf{d}^{t+1} = \operatorname{argmin}_{\{\mathbf{d} \in \mathcal{P}_N \,:\, \mathbf{d} \cdot \mathbf{u}^q \le \widehat{\gamma}_t, \text{ for } 1 \le q \le t\}} \Delta(\mathbf{d}, \mathbf{d}^1)$
   (e) If above infeasible or $\mathbf{d}^{t+1}$ contains a zero then $T = t$ and break
4. Output: $f_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{t=1}^T \alpha_t h^t(\mathbf{x})$, where the coefficients $\alpha_t$ maximize margin over hypotheses set $\{h^1, \ldots, h^T\}$.
```

Algorithm 3 TotalBoost$^g_\nu$ with accuracy parameter ν and edge guarantee g

As TotalBoost$_\nu$ but in step 3(c) we use $\widehat{\gamma}_t = g - \nu$.

and also the performance can very much depend on which LP solver is used (see experimental section).

Our algorithms are motivated by the minimum relative entropy principle of Jaynes: among the solutions satisfying some linear constraints choose the one that minimizes a relative entropy to the initial distribution $\mathbf{d}^1$, where the relative entropy is defined as follows: $\Delta(\widetilde{\mathbf{d}}, \mathbf{d}) = \sum_n \widetilde{d}_n \ln \frac{\widetilde{d}_n}{d_n}$. Our default initial distribution is uniform. However, the analysis works for any choice of $\mathbf{d}^1$ with non-zero components. There are two totally corrective versions of the algorithm: one that knows the guarantee g of the weak learner and one that does not. The one that does (called TotalBoost$^g_\nu$; Algorithm 3), simply constrains the edges of the previous hypotheses to be at most $g-\nu$, where ν is a given precision parameter. Our main algorithm, TotalBoost$_\nu$ (Algorithm 2) does not know g. It maintains the estimates $\widehat{\gamma}_t = \left(\min_{q=1}^t \gamma_q\right) - \nu$ and constrains the edges of the past hypotheses to be at most $\widehat{\gamma}_t$. The sequence $\{\widehat{\gamma}_t\}_t$ is clearly non-increasing. By our assumption $\gamma_t \ge g$, and therefore $\widehat{\gamma}_t \ge g - \nu$.

3. Termination Guarantees

When the algorithms break, we need to guarantee that the margin w.r.t. the current hypothesis set is at least

Algorithm 4 AdaBoost$^*_\nu$ with accuracy parameter ν

As TotalBoost$_\nu$ but minimize the divergence to the last distribution w.r.t. a single constraint:

$$\mathbf{d}^{t+1} = \operatorname*{argmin}_{\{\mathbf{d}:\mathbf{d}\cdot\mathbf{u}^t \le \widehat{\gamma}_t\}} \Delta(\mathbf{d}, \mathbf{d}^t).$$

Let $\widehat{\alpha}_t$ be the dual coefficient of the constraint on the edge of h^t used in iteration t. The algorithm breaks if the margin w.r.t. the current convex combination (i.e. the normalized $\widehat{\alpha}_t$) is at least $\widehat{\gamma}_t$.

Algorithm 5 AdaBoost$^g_\nu$ with accuracy parameter ν and guarantee g

As AdaBoost$^*_\nu$ but in step 3(c) we use $\widehat{\gamma}_t = g - \nu$.

$g-\nu$. TotalBoost$^g_\nu$ is given g and constrains the edges of all past hypotheses to be at most $g-\nu$. When these become infeasible, the edge γ^*_t w.r.t. the current hypotheses set is larger than $g-\nu$. The algorithm also breaks when the solution $\mathbf{d}^{t+1}$ of the minimization problem lies at the boundary of the simplex (i.e. the distribution has a zero component).[2] In this case $\gamma^*_t = g-\nu$, because if $\gamma^*_t < g-\nu$, then all constraints would have slack and the solution $\mathbf{d}$ that minimizes the divergence $\Delta(\mathbf{d}, \mathbf{d}^1)$ would lie in the interior of the simplex since $\mathbf{d}^1$ does. Thus whenever the algorithm breaks, we have $\rho^* - \nu \le \gamma^*_t$. TotalBoost$^g_\nu$ outputs a convex combination of the hypotheses $\{h^1, \dots, h^T\}$ that maximizes the margin. By duality, the value ρ^*_t of this margin equals the minimum edge γ^*_t and therefore TotalBoost$^g_\nu$ is guaranteed to output a combined hypothesis of margin larger than $g-\nu$.

The second algorithm TotalBoost$_\nu$ does not know the guarantee g of the weak learner. It breaks if its optimization problem becomes infeasible, which happens when $\gamma^*_t > \widehat{\gamma}_t \ge g-\nu$. The algorithm also breaks when the solution $\mathbf{d}^{t+1}$ of the minimization problem lies at the boundary of the simplex. In this case, $\gamma^*_t = \widehat{\gamma}_t$ by an argument similar to the one used above. Thus whenever the algorithm breaks, we have $\gamma^*_t \ge \widehat{\gamma}_t \ge g-\nu$ and therefore TotalBoost$_\nu$ is guaranteed to output a hypothesis of margin $\rho^*_t = \gamma^*_t \ge g-\nu$.

The termination condition for LPBoost[3] follows a similar argument: we directly check for $\gamma^*_t \ge \widehat{\gamma}_t$. The algorithm Adaboost$^*_\nu$ computes the margin using the normalized dual coefficients $\widehat{\alpha}_t$ of its constraints and stops as soon as this margin is at least $\widehat{\gamma}_t$. Finally, Adaboost$^g_\nu$ breaks when the same margin is at least $g-\nu$. For both of these algorithms the current distribution $\mathbf{d}^t$ lies in the interior because the dual coefficients $\widetilde{\alpha}_t$ are finite and $d^t_n \sim d^1_n \exp(-\sum_{q=1}^{t-1} \widetilde{\alpha}_q u^q_n)$.

[2]This second condition for breaking is only added to ensure the the dual variables of the optimization problem of TotalBoost$_\nu$ remain finite.

[3]We use a different termination condition for LPBoost than in (Bennett et al., 2000; Grove & Schuurmans, 1998).

4. Iteration Bound

In the previous section we showed that when the algorithms break, then the output hypothesis has margin at least $g-\nu$. We now show that TotalBoost$_\nu$ must break after $T \le \frac{2\ln N}{\nu^2}$ iterations. In each iteration t, the algorithm updates the distribution that is "closest" to $\mathbf{d}^1$ and lies in a certain convex set and these sets get smaller as t increases. Here closeness is measured with the relative entropy which is a special Bregman divergence. This closest point is called a *projection* of $\mathbf{d}^1$ to the convex set ($\mathbf{d}^1$ is assumed to lie in the interior of the simplex). The proof is analogous to an on-line mistake bound for learning disjunctions (Long & Wu, 2005). It employs the Generalized Pythagorean Theorem that holds for such projections w.r.t. any Bregman divergence (Bregman, 1967, Lemma 1; Herbster & Warmuth, 2001, Theorem 2).

Theorem 1 *TotalBoost$_\nu$ breaks after at most $\lceil \frac{2\ln N}{\nu^2} \rceil$ iterations.*

Proof Let $\mathcal{C}_t$ denote the convex set of all points $\mathbf{d} \in \mathbb{R}^N$ that satisfy $\sum_n d_n = 1$, $d_n \ge 0$ (for $1 \le n \le N$), and edge constraints $\mathbf{d}\cdot\mathbf{u}^q \le \widehat{\gamma}_t$, for $1 \le q \le t$, where $u^q_n = y_n h^q(\mathbf{x}_n)$. The distribution $\mathbf{d}^t$ at iteration $t-1$ is the projection of $\mathbf{d}^1$ onto the closed convex set $\mathcal{C}_{t-1}$. Notice that $\mathcal{C}_0$ is the entire simplex and because $\widehat{\gamma}_t$ can only decrease and a new constraint is added in trial t, we have $\mathcal{C}_t \subseteq \mathcal{C}_{t-1}$. If $t \le T-1$, then our termination condition assures that at trial $t-1$ the set $\mathcal{C}_{t-1}$ has a feasible solution in the interior of the simplex. Also $\mathbf{d}^1$ lies in the interior and $\mathbf{d}^{t+1} \in \mathcal{C}_t \subseteq \mathcal{C}_{t-1}$. These preconditions assure that at trial $t-1$ the projection $\mathbf{d}^t$ of $\mathbf{d}^1$ onto $\mathcal{C}_{t-1}$ exists and the Generalized Pythagorean Theorem for Bregman divergences can be applied:

$$\Delta(\mathbf{d}^{t+1}, \mathbf{d}^1) - \Delta(\mathbf{d}^t, \mathbf{d}^1) \ge \Delta(\mathbf{d}^{t+1}, \mathbf{d}^t). \tag{3}$$

Since $\mathbf{d}^t\cdot\mathbf{u}^t = \gamma_t$ and $\mathbf{d}^{t+1}\cdot\mathbf{u}^t \le \widehat{\gamma}_t \le \gamma_t - \nu$, $\mathbf{d}^t\cdot\mathbf{u}^t - \mathbf{d}^{t+1}\cdot\mathbf{u}^t \ge \nu$ and because $\mathbf{u}^t \in [-1,1]^N$, $|\mathbf{d}^{t+1} - \mathbf{d}^t|_1 \ge \nu$. We now apply Pinsker's inequality:

$$|\mathbf{d}^{t+1} - \mathbf{d}^t|_1 \ge \nu \text{ implies that } \Delta(\mathbf{d}^{t+1}, \mathbf{d}^t) > \frac{\nu^2}{2}. \tag{4}$$

By summing (3) over the first $T-1$ trials we obtain

$$\Delta(\mathbf{d}^T, \mathbf{d}^1) - \underbrace{\Delta(\mathbf{d}^1, \mathbf{d}^1)}_{0} > (T-1)\frac{\nu^2}{2}.$$

Since the left is at most $\ln N$, the bound of the theorem follows. ■

The key requirement for this proof is that the closed and convex constraint sets $\mathcal{C}_t$ used for the projection at trial t must be non-increasing. It is therefore easy to see that the iteration bound also holds for the TotalBoost$^g_\nu$ algorithm because of our assumption that $\gamma_t \geq g$. In the complete paper we prove the same iteration bound for corrective version AdaBoost$^*_\nu$, Adaboost$^g_\nu$, and the variants of TotalBoost where $\mathrm{argmin}(\mathbf{d}, \mathbf{d}^1)$ is replaced by $\mathrm{argmin}(\mathbf{d}, \mathbf{d}^t)$.

5. Experiments

In this section we illustrate the behavior of our new algorithms TotalBoost$_\nu$ and TotalBoost$^g_\nu$, and compare them with LPBoost and AdaBoost$^*_\nu$ on three different datasets:

- Dataset 1 is a public dataset from Telik Inc. for a drug discovery problem called COX-1: 125 binary labeled examples with a set of 3888 binary features that are complementation closed.
- Dataset 2 is an artificial dataset used in Rudin et al. (2004b) for investigating boosting algorithms that maximize the margin: 50 binary labeled examples with 100 binary features. For each original feature we added 99 similar features by inverting the feature value of one randomly chosen example (with replacement). This results in a 10,000 dimensional feature set of 100 blocks of size 100.
- Dataset 3 is a series of artificially generated datasets of 1000 examples with varying number of features but roughly constant margin. We first generated N_1 random ± 1-valued features $x_1, \ldots, x_{N_1}$ and set the label of the examples as $y = \mathrm{sign}(x_1 + x_2 + x_3 + x_4 + x_5)$. We then duplicated each features N_2 times, perturbed the features by Gaussian noise with $\sigma = 0.1$, and clipped the feature values so that they lie in the interval [-1,1]. We considered $N_1 = 1, 10, 100$ and $N_2 = 10, 100, 1000$.

The features of our datasets represent the values of the available weak hypotheses on the examples. In each iteration of boosting, the "base learner" simply selects the feature that maximizes the edge w.r.t. the current distribution $\mathbf{d}$ on the examples. This means that the guarantee g equals the maximum margin ρ^*. Note that our datasets and the base learner were chosen to exemplify certain properties of the algorithms and more extensive experiments are still needed.

We first discuss how the entropy minimization problems can be solved efficiently. We then compare the algorithms w.r.t. the number of iterations and the number of selected hypothesis. Finally we show how LPBoost is affected by the underlying optimizer and exhibit cases where LPBoost requires considerably more iterations than TotalBoost$_\nu$.

5.1. Solving the Entropy Problems

We use a "vanilla" *sequential quadratic programming* algorithm (Nocedal & Wright, 2000) for solving our main optimization problem:

$$\min_{\mathbf{d}\,:\,\sum_n d_n = 1,\ \mathbf{d} \geq \mathbf{0},\ \mathbf{u}^q \cdot \mathbf{d} \leq \widehat{\gamma}_t\ (1 \leq q \leq t)} \sum_{n=1}^{N} d_n \log \frac{d_n}{d_n^1}.$$

We initially set our approximate solution to $\widehat{\mathbf{d}} = \mathbf{d}^1$ and iteratively optimize $\widehat{\mathbf{d}}$. Given the current solution $\widehat{\mathbf{d}}$ satisfies the constraints $\sum_n \widehat{d}_n = 1$ and $\widehat{\mathbf{d}} \geq \mathbf{0}$, we determine an update $\boldsymbol{\delta}$ by solving the following problem:

$$\min_{\boldsymbol{\delta}} \left(\sum_{n=1}^{N} \left(1 + \log \frac{\widehat{d}_n}{d_n^1} \right) \delta_n + \frac{1}{2\widehat{d}_n} \delta_n^2 \right),$$

w.r.t. the constraints $\widehat{\mathbf{d}} + \boldsymbol{\delta} \geq \mathbf{0}$, $\sum_n \delta_n = 0$, and $\mathbf{u}^q \cdot (\widehat{\mathbf{d}} + \boldsymbol{\delta}) \leq \widehat{\gamma}_t$ (for $1 \leq q \leq t$). The estimate $\widehat{\mathbf{d}}$ is updated to $\widehat{\mathbf{d}} \leftarrow \widehat{\mathbf{d}} + \boldsymbol{\delta}$ and we repeat this process until convergence. The algorithms typically converges in very few steps.

Note that the above objective is the 2nd order Taylor approximation of the relative entropy $\Delta(\widehat{\mathbf{d}} + \boldsymbol{\delta}, \mathbf{d}^1)$ at $\boldsymbol{\delta} = \mathbf{0}$. The resulting optimization problem is quadratic with a diagonal Hessian and can be efficiently solved by off-the-shelf optimizer packages (e.g. ILOG CPLEX).

5.2. Number of Iterations

First, we consider the number of iterations needed until each of the algorithms has achieved a margin of at least $\rho^* - \nu$. We use dataset 1 and record the margin of the convex combination of hypotheses produced by TotalBoost$_\nu$, LPBoost and AdaBoost$^*_\nu$. Additionally, we compute the maximal margin of the current hypothesis sets in each iteration. See Figure 1 for details. The default optimizer used for solving LPs and QPs is ILOG CPLEX's interior point method.

It should be noted that AdaBoost$^*_\nu$ needs considerably less computations per iteration than the totally corrective algorithms. In the case where calling the base learner is very cheap, AdaBoost$^*_\nu$ may in some unusual cases require less computation time than TotalBoost$_\nu$. However, in our experiments, the number of iterations required by AdaBoost$^*_\nu$ to achieve margin at

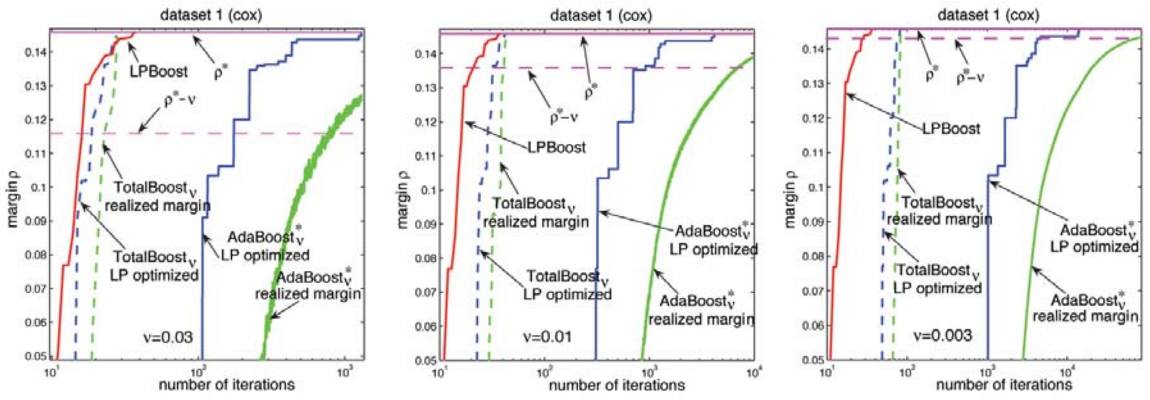

Figure 1: TotalBoost$_\nu$, LPBoost and AdaBoost$^*_\nu$ on dataset 1 for $\nu = 0.03, 0.01, 0.003$: We show the margin realized the normalized dual coefficients $\widehat{\alpha_t}$ of TotalBoost$_\nu$ and AdaBoost$^*_\nu$ (green) and the LP-optimized margin ρ^*_t (1) (blue). Observe that AdaBoost$^*_\nu$ needs several thousands iterations and the number of iterations of TotalBoost$_\nu$ and LPBoost are comparable. The margins of TotalBoost$_\nu$ and AdaBoost$^*_\nu$ start growing slowly, in particular when ν is small. The margin of TotalBoost$^g_\nu$ (with guarantee $g = \rho^*$) increases faster than LPBoost (not shown).

least $\rho^* - \nu$ was $\approx$ 1/10 times the theoretical upper bound $2\log(N)/\nu^2$. TotalBoost$_\nu$ typically requires much fewer iterations, even though no improved theoretical bound is known for this algorithm. In our experience, the iteration number of TotalBoost$_\nu$ depends only slightly on the precision parameter ν and when $\widehat{\gamma}_t$ is close to ρ^*, then this algorithm converges very fast to the maximum margin solution (LPBoost has a similar behavior).

While the algorithms AdaBoost$^*_\nu$ and TotalBoost$_\nu$ provably maximize the margin, they both have the problem of starting too slowly for small ν. If there is any good upper bound available for the guarantee g (which here is the optimal margin ρ^*), then we can initialize $\widehat{\gamma}_t$ with this upper bound and speed up the starting phase. In particular, when ρ^* is known exactly, then the algorithms AdaBoost$^g_\nu$ and TotalBoost$^g_\nu$ require drastically fewer iterations and the latter consistently beats LPBoost (not shown). In practical situations it is often easy to obtain a reasonable upper bound for g.

5.3. Number of Hypotheses

In this subsection, we compare how many hypotheses the algorithms need to achieve a large margin. Note that LPBoost and TotalBoost$_\nu$ only *select* a base hypothesis once: After the first selection, the distribution $\mathbf{d}$ is maintained such that the edge for that hypothesis is smaller than $\widehat{\gamma}_t$ and it is not selected again. AdaBoost$^*_\nu$ may select the same hypothesis many times. However, if there are several similar features (as in datasets 2 & 3), then this corrective algorithm often selects hypotheses that are similar to previously selected ones and the number of weak hypotheses used in the final convex combination is unnecessarily large. Hence, TotalBoost$_\nu$ and LPBoost seem better suited for feature selection, when small ensembles are needed.

In Figure 2 we display the margin vs. the number of *used* and *selected* hypotheses. The number of selected hypothesis for LPBoost and TotalBoost$_\nu$ is equal to the number of iterations. For these algorithms a previously selected hypothesis can become inactive (corresponding $\alpha = 0$). In this case it is not counted as a used hypothesis. Note that the number of used hypotheses for LPBoost may depend on the choice of the optimizer (also see discussion below). In the case of AdaBoost$^*_\nu$, all dual coefficients $\widehat{\alpha_t}$ are non-zero in the final convex combination. (See caption of Figure 2 for more details.) We can conclude that the totally corrective algorithms need considerable less hypotheses when there are many redundant hypotheses/features. LPBoost and TotalBoost$_\nu$ differ in the initial iterations (depending on ν), but produce combined hypotheses of similar size.

In Figure 3 we compare the effect of different choices of the optimizer for LPBoost. For dataset 2 there is a surprisingly large difference between interior point and simplex based methods. The reason is that the weights computed by the simplex method are often sparse and the changes in the duplicated features are sparse as well (by design). Hence, it can easily happen that the base learner is “blind” on some examples when selecting the hypotheses. Interior point methods find a solution in the interior and therefore distribute the weights among the examples. To illustrate that this is the right explanation, we modify LPBoost such that it first computes γ^*_t but then it computes the

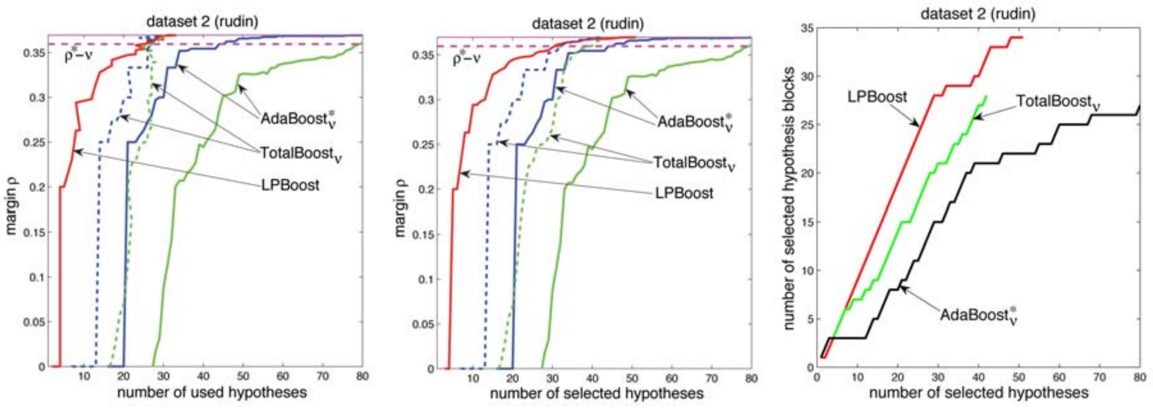

Figure 2: TotalBoost$_\nu$, LPBoost and AdaBoost$^*_\nu$ on dataset 2 for $\nu = 0.01$: [*left & middle*] The realized (green) and the LP-optimized (blue) margin ρ^*_t (as in Figure 1) vs. the number of used (active) and selected (active or inactive) hypotheses in the convex combination. We observe that the totally corrective algorithms use considerable less hypotheses than the AdaBoost$^*_\nu$. If $\nu \ll 0.01$, then TotalBoost$_\nu$ is again affected by the slow start which leads to a relatively large number of selected hypotheses in the beginning. [*right*] The number of selected hypotheses vs. the number of selected blocks of hypotheses. AdaBoost$^*_\nu$ often chooses additional hypotheses from previously chosen blocks, while LPBoost typically uses only one per block and TotalBoost$_\nu$ a few per block. When $\nu = .1$, TotalBoost$_\nu$ behaves more like LPBoost (not shown).

weights using the relative entropy minimization with $\widehat{\gamma}_t = \gamma^*_t + \epsilon$ (where $\epsilon = 10^{-4}$). We call this the regularized LPBoost algorithm. We observe in Figure 3 that the regularization considerably improves the convergence speed to ρ^* of the simplex based solver.

5.4. Redundancy in High Dimensions

We found that LPBoost usually performs very well and is very competitive to TotalBoost$_\nu$ in terms of the number of iterations. Additionally, it only needs to solve linear and not entropy minimization problems. However, no iteration bound is known for LPBoost that is independent of the size of the hypothesis set. We performed a series of experiments with increasing dimensionality and compared LPBoost's and TotalBoost$_\nu$'s convergence speed. We found that in rather high dimensional cases, LPBoost converges quite slowly when features are redundant (see Figure 4 for an example using dataset 3). In future work, we will investigate why LPBoost converges more slowly in this example and construct more extreme datasets that show this.

6. Conclusion

We view boosting as a relative entropy projection method and obtain our iteration bounds without bounding the average training error in terms of the product of exponential potentials as is customarily done in the boosting literature (see e.g. Schapire and Singer (1999)). In the full paper we will relate our methods to the latter slightly longer proof style.

The proof technique based on Bregman projection and the Generalized Pythagorean theorem is very versatile. The iteration bound of $O(\frac{\log N}{\nu^2})$ holds for all boosting algorithms that use constrained minimization of any Bregman divergence $\widetilde{\Delta}(.,.)$ over a domain that contains the probability simplex for which $\inf_{\mathbf{d}\in\mathcal{C}_t} \widetilde{\Delta}(\mathbf{d},\mathbf{d}^t) = \Omega(\nu^2)$ and $\widetilde{\Delta}\left(\mathbf{d}^T, (\frac{1}{N})\right) = O(\log N)$. For example, the sum of binary entropies Δ_2 has both these properties:

$$\inf_{\mathcal{C}_t} \overbrace{\sum_n \left(d_n \ln \frac{d_n}{d^t_n} + (1-d_n)\ln\frac{1-d_n}{1-d^t_n}\right)}^{:=\Delta_2(\mathbf{d},\mathbf{d}^t)}$$
$$\geq \inf_{\mathcal{C}_t} \Delta(\mathbf{d},\mathbf{d}^t) + \underbrace{\inf_{\mathbf{d}:\sum_n d_n = 1} \Delta(\mathbf{1}-\mathbf{d},\mathbf{1}-\mathbf{d}^t)}_{0} \overset{(4)}{\geq} \frac{\nu^2}{2},$$

where the first inequality follows from splitting the inf and dropping one of the constraints from the constraint set $\mathcal{C}_t$ and $\mathbf{1}$ denotes the all one vector. Furthermore, $\Delta_2\left(\mathbf{d}^{T-1}, (\frac{1}{N})\right) \leq (\ln N)+1$ and this leads to an iteration bound of $\frac{2((\ln N)+1)}{\nu^2}$. The corrective version based on this divergence has been called *LogitBoost* (Friedman et al., 2000; Duffy & Helmbold, 2000). The above reasoning immediately provides $O(\frac{\log N}{\nu^2})$ iteration bounds for the totally corrective versions of LogitBoost that maximize the margin. Even though the theoretical bounds for the LogitBoost variants are essentially the same as the bounds for the standard relative entropy algorithms discussed in this paper, the LogitBoost variants are marginally inferior in practice (not shown).

Both the corrective and totally corrective algorithms for maximizing the margin start rather slowly and heuristics are needed for decreasing the edge bound $\widehat{\gamma}_t$

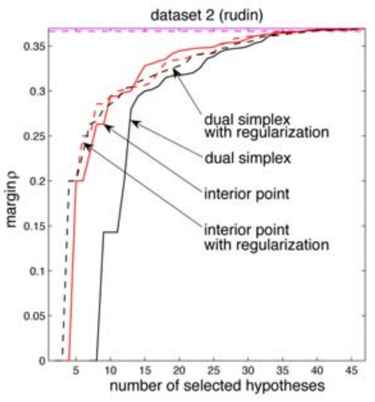

Figure 3: LPBoost with different optimizers: shown is the margin vs. the no. of selected hypotheses. Different optimizers lead to the selection of different hypotheses with varying maximum margins. Adding a regularizer (see text) significantly improves the simplex solution in some cases.

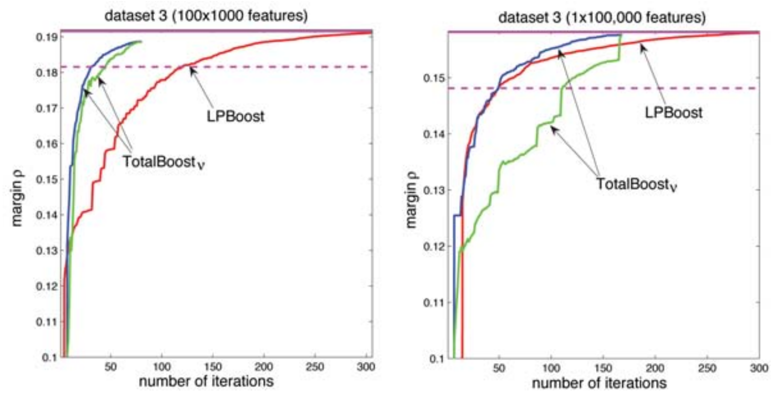

Figure 4: LPBoost vs. TotalBoost$_\nu$ on two 100,000 dimensional datasets. Shown is the margins vs. the number of iterations: [*left*] data with 100 duplicated blocks (with clipped Gaussian noise) and [*right*] data with independent features. For TotalBoost$_\nu$, we depict the realized (green) and the LP-optimized (blue) margin. When there are lots of duplicated features, then LPBoost stalls after an initial fast phase, while it performs well in other cases. We did not observe this behavior for TotalBoost$_\nu$ or AdaBoost$^*_\nu$ (not shown). The difference becomes larger when the block size is increased.

so that this slow start is avoided. For practical noisy applications, boosting algorithms are needed that allow for a bias term and for soft margins. LPBoost has already been used this way in Bennett et al. (2000) but no iteration bounds are known for any version of LPBoost. We show in the full paper that our methodology still leads to iteration bounds for boosting algorithms with entropic regularization when a bias term is added. Iteration bounds for soft margin versions are left as future research.

References

Bennett, K., Demiriz, A., & Shawe-Taylor, J. (2000). A column generation algorithm for boosting. *Proc. ICML* (pp. 65–72). Morgan Kaufmann.

Bregman, L. (1967). The relaxation method for finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Math. and Math. Physics*, *7*, 200–127.

Breiman, L. (1997). *Prediction games and arcing algorithms*Technical Report 504). Statistics Department, University of California at Berkeley.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, *11*, 1493–1518.

Duffy, N., & Helmbold, D. (2000). Potential boosters? *NIPS'00* (pp. 258–264).

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. & Sys. Sci.*, *55*, 119–139.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive Logistic Regression: a statistical view of boosting. *Annals of Statistics*, *2*, 337–374.

Grove, A., & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. *Proc. 15th Nat. Conf. on Art. Int.*.

Herbster, M., & Warmuth, M. (2001). Tracking the best linear prediction. *J. Mach. Learn. Res.*, 281–309.

Kivinen, J., & Warmuth, M. (1999). Boosting as entropy projection. *COLT'99*.

Lafferty, J. (1999). Additive models, boosting, and inference for generalized divergences. *COLT'99* (pp. 125–133).

Littlestone, N. (1988). Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Long, P. M., & Wu, X. (2005). Mistake bounds for maximum entropy discrimination. *NIPS'04* (pp. 833–840).

Nocedal, J., & Wright, S. (2000). *Numerical optimization.* Springer Series in Op. Res. Springer.

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

Rätsch, G., & Warmuth, M. K. (2005). Efficient margin maximizing with boosting. *J. Mach. Learn. Res.*, 2131–2152.

Rudin, C., Daubechies, I., & Schapire, R. (2004a). Dynamics of AdaBoost: Cyclic behavior and convergence of margins. *J. Mach. Learn. Res.*, 1557–1595.

Rudin, C., Schapire, R., & Daubechies, I. (2004b). Analysis of boosting algoritms using the smooth margin function: A study of three algorithms. Unpublished manuscript.

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

Schapire, R., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*, 297–336.