Boosting with Structural Sparsity

John Duchi JDUCHI@CS.BERKELEY.EDU
Department of EECS, University of California, Berkeley, CA 94720

Yoram Singer SINGER@GOOGLE.COM
Google, Mountain View, CA 94043

Abstract

We derive generalizations of AdaBoost and related gradient-based coordinate descent methods that incorporate sparsity-promoting penalties for the norm of the predictor that is being learned. The end result is a family of coordinate descent algorithms that integrate forward feature induction and back-pruning through regularization and give an automatic stopping criterion for feature induction. We study penalties based on the ℓ_1, ℓ_2, and ℓ_∞ norms of the predictor and introduce mixed-norm penalties that build upon the initial penalties. The mixed-norm regularizers facilitate structural sparsity in parameter space, which is a useful property in multiclass prediction and other related tasks. We report empirical results that demonstrate the power of our approach in building accurate and structurally sparse models.

1. Introduction and problem setting

Boosting is a highly effective and popular method for obtaining an accurate classifier from a set of inaccurate predictors. Boosting algorithms construct high precision classifiers by taking a weighted combination of base predictors, known as weak hypotheses (see Meir and Rätsch (2003) and the numerous references therein). Many boosting algorithms can also be viewed as forward-greedy feature induction procedures. In this view, the weak-learner provides new predictors which perform well either in terms of their error-rate with respect to the distribution that boosting maintains or in terms of their potential in reducing the empirical loss. Once a feature is chosen, typically in a greedy manner, it is associated with a weight which remains intact through the reminder of the boosting process.

The aesthetics and simplicity of AdaBoost and other forward greedy algorithms, such as LogitBoost (Friedman et al., 2000), also facilitate a tacit defense from overfitting, especially when combined with early stopping of the boosting process (Zhang & Yu, 2005). The empirical success of Boosting algorithms helped popularize the view that boosting algorithms are resilient to overfitting. However, several researchers have noted the deficiency of forward-greedy boosting algorithms and suggest alternative coordinate descent algorithms, such as totally-corrective boosting (Warmuth et al., 2006) or Zhang's forward/backward algorithms (2008). The algorithms that we present in this paper build on existing boosting and other coordinate descent procedures while incorporating, throughout their run, regularization of the weights of the selected features. The added regularization terms influence both the selection of new features and the weight assignment steps. Moreover, as we discuss below, the regularization may eliminate (by assigning a weight of zero) previously selected features. The result is a simple procedure that includes forward induction, weight estimation, backward pruning, entertains convergence guarantees, and yields sparse models. We are also able to group features and impose structural sparsity on the learned weights, which is a focus and one of the main contributions of this paper.

Our starting point is a simple yet effective modification to AdaBoost that incorporates an ℓ_1 penalty for the norm of the weight vector it constructs. The update we devise can be used both for weight optimization as well as for induction of new accurate hypotheses while taking the resulting 1-norm into account. A closely related approach was suggested by Dudík et al. (2007) in the context of maximum entropy, though our analysis for classification is a special case of an abstract saddle-point theorem that we prove. This general theorem is also applicable to other norms and losses, in particular the ℓ_∞ norm, which serves as a building block for imposing structural sparsity.

For simplicity, we assume that the class of weak hypotheses is finite and contains n different hypotheses. We thus map each instance $\boldsymbol{x}$ to an n dimensional vector $(h_1(\boldsymbol{x}), \ldots, h_n(\boldsymbol{x}))$, and we overload notation and simply denote the vector as $\boldsymbol{x} \in \mathbb{R}^n$. Though omitted due to the lack of space, our framework can be used in the presence of countably infinite features, also known as the

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009.

task of feature induction. Each instance $\boldsymbol{x}_i$ is associated with a label $y_i \in \{-1, +1\}$. The goal of learning is then to find a weight vector $\boldsymbol{w}$ such that the sign of $\boldsymbol{w} \cdot \boldsymbol{x}_i$ is equal to y_i. Moreover, we would like to attain large inner-products so long as the predictions are correct. Formally, given a sample $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, the algorithm focuses on finding $\boldsymbol{w}$ for which the empirical logistic loss, $L(\boldsymbol{w}) = \sum_{i=1}^m \log\left(1 + \exp(-y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i)\right)$, is small. Our derivation also applies to the exp-loss. We first adapt AdaBoost to incorporate the ℓ_1-norm of the weight vector into the empirical loss, $\sum_{i=1}^m \log\left(1 + \exp(-y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i))\right) + \lambda\|\boldsymbol{w}\|_1$.

This problem is by no means new. It is often referred to as ℓ_1-regularized logistic regression and several advanced optimization methods have been designed for the problem (Koh et al., 2007). ℓ_1-regularization has many advantages, including its ability to yield sparse weight vectors $\boldsymbol{w}$ and, under certain conditions, to recover the true sparsity of $\boldsymbol{w}$ (Zhao & Yu, 2006). We extend this by replacing the ℓ_1-norm with a mixed-norm regularizer (denoted ℓ_1/ℓ_p) to achieve structural sparsity. Mixed-norm regularization is used when there is a partition or structure over the weights that separates them into disjoint groups of parameters, and an ℓ_p-norm ties each group. For concreteness and in order to leverage existing boosting algorithms, we specifically focus on settings in which we have a matrix $W = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_k] \in \mathbb{R}^{n \times k}$ of weight vectors, and we regularize the weights in each row of W (denoted $\overline{\boldsymbol{w}}_j$) together through an ℓ_p-norm. We derive updates for two important settings that generalize binary logistic-regression. The first is multitask learning (Obozinski et al., 2007), in which we have a set of tasks $\{1, \ldots, k\}$ and a weight vector $\boldsymbol{w}_k$ for each task. Our goal is to learn a matrix W minimizing

$$\sum_{i=1}^{m}\sum_{r=1}^{k} \log\left(1 + e^{-y_{i,r}(\boldsymbol{w}_r \cdot \boldsymbol{x}_i)}\right) + \lambda \sum_{j=1}^{n} \|\overline{\boldsymbol{w}}_j\|_p \quad . \quad (1)$$

The other generalization we describe is the multiclass logistic loss. We again assume there are k weight vectors that operate on each instance. Given an example $\boldsymbol{x}_i$, the classifier's prediction is a vector $[\boldsymbol{w}_1 \cdot \boldsymbol{x}_i, \ldots, \boldsymbol{w}_k \cdot \boldsymbol{x}_i]$, and the predicted class is the index of the inner-product attaining the largest of the k values, $\operatorname{argmax}_r \boldsymbol{w}_r \cdot \boldsymbol{x}_i$. In this case, the regularized loss $Q(W)$ is

$$\sum_{i=1}^{m} \log\left(1 + \textstyle\sum_{r \neq y_i} e^{\boldsymbol{w}_r \cdot \boldsymbol{x}_i - \boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i}\right) + \lambda \sum_{j=1}^{n} \|\overline{\boldsymbol{w}}_j\|_p \quad (2)$$

We also give a new upper bound for the multiclass loss. Previous multiclass constructions for boosting assume that each base hypothesis provides a different prediction per class, so they are not directly applicable to the multiclass setting discussed in this paper, which allocates a dedicated predictor per class. Our result is an efficient multiclass and multitask boosting-based procedure with ℓ_1/ℓ_∞ regularization. We then shift our focus to an alternative apparatus for coordinate descent with the log-loss that does not stem from the AdaBoost algorithm. In this approach we upper bound the log-loss by a quadratic function, terming the resulting update GradBoost as it updates coordinates in a fashion that follows the gradient. Similar to the generalization of AdaBoost, we study ℓ_1 and ℓ_∞ penalties by *reusing* the fixed-point theorem, and we derive GradBoost updates for both ℓ_1/ℓ_∞ and ℓ_1/ℓ_2 mixed-norm regularization.

It is clearly impossible to cover related work in depth and we give here a very brief overview. Our derivation follows the template-based algorithm of Collins et al. (2002), incorporating regularization in a way analogous to Dudík et al.'s maximum-entropy framework (2007). The base GradBoost algorithm we derive shares similarity with LogitBoost (Friedman et al., 2000) while our bounding technique was first suggested by Dekel et al. (2005). Learning sparse models through ℓ_1 regularization is the focus of a voluminous amount of work in different research areas, from statistics to information theory (Zhao & Yu, 2006; Koh et al., 2007; Zhang, 2008). Multiple authors have studied the setting of mixed-norm regularization, which is of great interest in the statistical estimation literature, though the focus is typically on consistency for linear regression rather than efficient algorithms. Negahban and Wainwright (2008) recently analyzed sparsity through ℓ_1/ℓ_∞ mixed-norms, and Obozinski et al. (2007) analyze ℓ_1/ℓ_2.

2. AdaBoost with ℓ_1 regularization

We now outline our ℓ_1-infused modification to AdaBoost, providing only a sketch, since the algorithm can be obtained as a special case of the analysis presented in Sec. 3. We build on existing analyses of AdaBoost, which all derive upper bounds on the loss which the booster then minimizes. We can then rely on the fact that each round of boosting is guaranteed to decrease the penalized loss. In the generalized version of boosting (Collins et al., 2002), the booster selects a vector $\boldsymbol{a}$ from a set of templates $\mathcal{A}$ on each round of boosting. The template selects the set of base hypotheses whose weight we update. Moreover, the template vector can specify a different budget for each feature update so long as the vector $\boldsymbol{a}$ satisfies the condition $\sum_j a_j |x_{i,j}| \leq 1$. Classical boosting sets a single coordinate in the vector $\boldsymbol{a}$ to a non-zero value. We start by recalling the progress bound for AdaBoost with the log-loss when using a template vector. Define importance weights $q^t(i) = 1/(1 + e^{y_i \boldsymbol{w}^t \cdot \boldsymbol{x}_i})$, which are the probability the current classifier assigns to the incorrect label for example i, and weighted correlations

$$\mu_j^+ = \sum_{i : y_i x_{i,j} > 0} q^t(i)|x_{i,j}| \; ; \; \mu_j^- = \sum_{i : y_i x_{i,j} < 0} q^t(i)|x_{i,j}| \quad .$$

Let $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{\delta}^t$, $\delta_j^t = a_j d_j^t$, and $\boldsymbol{a}$ satisfy $\sum_j a_j |x_{i,j}| \leq 1$. Then the change in the log-loss, $\Delta_t = L(\boldsymbol{w}^t) - L(\boldsymbol{w}^{t+1})$, between two iterations of boosting is lower bounded by (Collins et al., 2002)

$$\Delta_t \geq \sum_{j=1}^{n} a_j \left(\mu_j^+ \left(1 - e^{-d_j^t}\right) + \mu_j^- \left(1 - e^{d_j^t}\right) \right) \ .$$

Since the ℓ_1 penalty is an additive term, we incorporate the change in the 1-norm of $\boldsymbol{w}$ to bound the overall decrease in the loss when updating $\boldsymbol{w}^t$ to $\boldsymbol{w}^t + \boldsymbol{\delta}^t$ with $\Delta_t - \lambda \|\boldsymbol{\delta}^t + \boldsymbol{w}^t\|_1 + \lambda \|\boldsymbol{w}^t\|_1$. By construction, this bound is additive in δ_j. Thus, we omit the index j, eliminate constants, and are left with the following minimization problem in δ:

$$\min_{\delta} \ a\mu^+ e^{-\delta/a} + a\mu^- e^{\delta/a} + \lambda \left|\delta + w\right| \ . \qquad (3)$$

We state two lemmas that aid us in finding the optimum of Eq. (3). The lemmas are special cases of Thm. 1 later.

Lemma 2.1. *If $\mu^+ e^{w/a} - \mu^- e^{-w/a} > 0$, then the minimizing $\delta^\star$ of Eq. (3) satisfies $\delta^\star + w \geq 0$. Likewise, if $\mu^+ e^{w/a} - \mu^- e^{-w/a} < 0$, then $\delta^\star + w \leq 0$.*

Lemma 2.2. *The solution of Eq. (3) with respect to δ is $\delta^\star = -w$ if and only if $\left|\mu^+ e^{w/a} - \mu^- e^{-w/a}\right| \leq \lambda$.*

Equipped with the above lemmas, the update to w_j^{t+1} is straightforward to derive. Let us assume without loss of generality that $\mu_j^+ e^{w_j^t/a_j} - \mu_j^- e^{-w_j^t/a_j} > \lambda$, so that $\delta_j^\star \neq -w_j$ and $\delta_j^\star + w_j > 0$. We need to solve the equation $-\mu_j^+ e^{-\delta_j/a_j} + \mu_j^- e^{\delta_j/a_j} + \lambda = 0$ or $\mu_j^- \beta^2 + \lambda\beta - \mu_j^+ = 0$, where $\beta = e^{\delta_j/a_j}$. Since β is strictly positive, it is equal to the positive root of the above quadratic equation, yielding

$$\delta_j^\star = a_j \log\left(-\lambda + \sqrt{\lambda^2 + 4\mu_j^+\mu_j^-}/(2\mu_j^-) \right) \ . \qquad (4)$$

In the symmetric case, when $\delta_j^\star + w_j^t < 0$, we get that $\delta_j^\star = a_j \log(\lambda + \sqrt{\lambda^2 + 4\mu_j^+\mu_j^-}/(2\mu_j^-))$. Finally, when $|\mu_j^+ e^{w_j^t/a_j} - \mu_j^- e^{-w_j/a_j}| \leq \lambda$, Lemma 2.2 implies that $\delta_j^\star = -w_j^t$. When μ_j^- is zero and $\mu_j^+ e^{w_j^t/a_j} > \lambda$ the solution is $\delta_j^\star = a_j \log(\mu_j^+/\lambda)$, and analogously for $\mu_j^+ = 0$.

3. Incorporating ℓ_∞ regularization

We now begin to lay the framework for multitask and multiclass boosting with mixed-norm regularizers. The main theorem in this section can be used to derive and analyze the algorithm of previous section. Before deriving updates for boosting, we consider a more general framework of minimizing a sum of one-dimensional, convex, bounded below and differentiable functions $f_j(d)$ plus an ℓ_∞-regularizer. We want to solve

$$\min_{\boldsymbol{d}} \sum_{j=1}^{k} f_j(d_j) + \lambda \left\|\boldsymbol{d}\right\|_\infty \ . \qquad (5)$$

The following theorem characterizes the solution $\boldsymbol{d}^\star$ of Eq. (5), and leads to efficient algorithms for specific functions f_j. We use $[k]$ as a shorthand for the set $\{1, 2, \ldots, k\}$.

Theorem 1. *Let $\tilde{d}_j$ satisfy $f_j'(\tilde{d}_j) = 0$. The optimal solution $\boldsymbol{d}^\star$ of Eq. (5) satisfies the following properties:*

(i) $\boldsymbol{d}^\star = \boldsymbol{0}$ if and only if $\sum_{j=1}^k |f_j'(0)| \leq \lambda$.

(ii) For all j, $f_j'(0) d_j^\star \leq 0$ and $f_j'(d_j^\star) d_j^\star \leq 0$.

(iii) Let $B = \left\{j : |d_j^\star| = \left\|\boldsymbol{d}^\star\right\|_\infty\right\}$ and $U = [k] \setminus B$:

(a) For all $j \in U$, $\tilde{d}_j = d_j^\star$ and $f_j'(d_j^\star) = 0$.

(b) For all $j \in B$, $|\tilde{d}_j| \geq |d_j^\star| = \left\|\boldsymbol{d}^\star\right\|_\infty$

(c) When $\boldsymbol{d}^\star \neq \boldsymbol{0}$, $\sum_{j \in B} |f_j'(d_j^\star)| = \lambda$.

Sketch of Proof We sketch a proof of the theorem using subgradient calculus (Bertsekas, 1999). For point (i), we note that the subgradient set of $\lambda \left\|\boldsymbol{d}\right\|_\infty$ evaluated at $\boldsymbol{d} = \boldsymbol{0}$ is the set of vectors $\{\boldsymbol{z} : \left\|\boldsymbol{z}\right\|_1 \leq \lambda\}$. Thus, we look at the ℓ_1-norm of the gradient of the sum of functions at $\boldsymbol{d} = \boldsymbol{0}$, and if $\sum_{j=1}^k |f_j'(0)| \leq \lambda$ then $\boldsymbol{d}^\star = \boldsymbol{0}$ and vice versa. Point (ii) is a consequence of the monotonicity of the derivatives (or subgradient sets) of convex functions (the derivatives are non-decreasing). For point (iii), we note that if $\boldsymbol{d}^\star = \boldsymbol{0}$, then (a), (b), and (c) are trivially satisfied. If not, then consider the subgradient set of the ℓ_∞ norm:

$$\partial\lambda \left\|\boldsymbol{d}\right\|_\infty = \lambda \operatorname{Co}\left\{\operatorname{sign}(d_i)\boldsymbol{e}_i : |d_i| = \left\|\boldsymbol{d}\right\|_\infty\right\} \ .$$

For part (a), any $j \in U$ must have derivative $f_j'(d_j^\star) = 0$ to satisfy the subgradient conditions for optimality. For part (b), we note that if $j \in B$ and $|\tilde{d}_j| < |d_j^\star|$, we can move $d_j^\star$ toward $\tilde{d}_j$ while decreasing or not changing $\left\|\boldsymbol{d}^\star\right\|_\infty$, and we decrease $f_j(d_j)$. Part (c) is another consequence of the subgradient conditions for optimality. □

In Fig. 1, we present a general algorithm for minimizing $\sum_j f_j(d_j) + \lambda \left\|\boldsymbol{d}\right\|_\infty$ that builds directly on Thm. 1. The algorithm flips signs so that all $d_j \geq 0$ (see part (ii) of the theorem). It then iteratively adds points to the bounded set B, starting from the point with largest unregularized solution $\tilde{d}_{(1)}$ (see part (iii)). When the algorithm finds a set B and bound $\xi = \left\|\boldsymbol{d}\right\|_\infty$ so that part (iii) of the theorem is satisfied (which is guaranteed by (iii.b)), it terminates. The algorithm naïvely has runtime complexity $O(k^2)$, which we bring down to $O(k \log k)$ in the sequel by exploiting the specific structure of f_j.

Revisiting AdaBoost with ℓ_1-regularization Lemmas 2.1 and 2.2 are special cases of the theorem. Recall the ℓ_1-regularized minimization problem we faced for the exponential loss in Eq. (3). We had to minimize a function of the form $a\mu^+ e^{-\delta/a} + a\mu^- e^{\delta/a} + \lambda|\delta + w|$. Replacing $\delta + w$ with θ, this minimization problem is equivalent to minimizing $a\mu^+ e^{w/a} e^{-\theta/a} + a\mu^- e^{-w/a} e^{\theta/a} + \lambda|\theta|$, a one dimensional version of the problem considered in Thm. 1.

INPUT: Functions $\{f_r\}_{r=1}^k$, regularization λ
IF $\sum_{r=1}^k |f_r'(0)| \leq \lambda$
  RETURN $\boldsymbol{d}^\star = \mathbf{0}$
// Find sign of optimal solutions
SET $s_r = -\operatorname{sign}(f_r'(0))$
// Get ordering of unregularized solutions
SOLVE $\tilde{d}_r = \operatorname{argmin}_d f_r(s_r d)$
// We have $\tilde{d}_{(1)} \geq \tilde{d}_{(2)} \geq \ldots \geq \tilde{d}_{(k)}$
SORT $\{\tilde{d}_r\}$ (descending) into $\{\tilde{d}_{(r)}\}$; $\tilde{d}_{(k+1)} = 0$
FOR $l = 1$ to k
  SOLVE for ξ such that $\sum_{i=1}^l f'_{(i)}(s_i \xi) = -\lambda$
  IF $\xi \geq \tilde{d}_{(l+1)}$
    BREAK
RETURN $\boldsymbol{d}^\star$ such that $d_r^\star = s_r \min\{\tilde{d}_r, \xi\}$

Figure 1. Algorithm for minimizing $\sum_r f_r(d_r) + \lambda \|\boldsymbol{d}\|_\infty$.

4. AdaBoost with ℓ_1/ℓ_∞ regularization

In this section we present generalizations of AdaBoost to the multitask and multiclass problems with mixed-norm regularization given in Eq. (1) and Eq. (2). We start by deriving boosting-style updates for the mixed-norm *multi-task* loss of Eq. (1) with $p = \infty$. The multitask loss is decomposable into sums of losses, one per task. Hence, for each separate task we obtain the same bound as that in the binary classification case. However, we now need to update rows $\overline{\boldsymbol{w}}_j$ from the matrix W while taking into account the mixed-norm penalty. Given a row j, we calculate importance weights $q^t(i,r)$ for each example i and task r as the probability the current weights assign the incorrect label, $q^t(i,r) = 1/(1+\exp(y_{i,r}\boldsymbol{w}_r \cdot \boldsymbol{x}_i))$, and the correlations $\mu^{\pm}_{r,j}$ for each task r as $\mu^+_{r,j} = \sum_{i:y_{i,r}x_{i,j}>0} q^t(i,r)|x_{i,j}|$ and $\mu^-_{r,j} = \sum_{i:y_{i,r}x_{i,j}<0} q^t(i,r)|x_{i,j}|$. Defining $\boldsymbol{\delta}_j = [\delta_{j,1} \cdots \delta_{j,k}]$ and applying the progress bound for binary classification, we see when we perform the update $W^{t+1} = W^t + [\boldsymbol{\delta}_1^t \cdots \boldsymbol{\delta}_n^t]^\top$ we can lower bound the change in the loss, $\Delta_t = L(W^{t+1}) - L(W^t)$, with

$$\sum_{j,r} a_j \left(\mu^+_{r,j}(1 - e^{-\delta^t_{j,r}/a_j}) + \mu^-_{r,j}(1 - e^{\delta^t_{j,r}/a_j})\right) \quad .$$

As before, the template vector should satisfy the constraint that $\sum_j a_j |x_{i,j}| \leq 1$ for all i.

For the multiclass objective from Eq. (2), we change the definition of the importance weights and correlations $\mu^{\pm}_{r,j}$, which gives us a new bound on the change in the multiclass loss. The following lemma extends the boosting bounds of Collins et al. (2002).

Lemma 4.1 (Multiclass boosting progress bound). *Let $q^t(i,r)$ denote the importance weight for each example index i and class index $r \in \{1,\ldots,k\}$, where*

$$q^t(i,r) = \frac{\exp(\boldsymbol{w}_r^t \cdot \boldsymbol{x}_i)}{\sum_l \exp(\boldsymbol{w}_l^t \cdot \boldsymbol{x}_i)} \quad .$$

Define the importance-weighted correlations as

$$\mu^+_{r,j} = \sum_{i:y_i \neq r, x_{i,j}<0} q^t(i,r)|x_{i,j}| + \sum_{i:y_i=r, x_{i,j}>0} (1-q^t(i,y_i))|x_{i,j}|$$
$$\mu^-_{r,j} = \sum_{i:y_i \neq r, x_{i,j}>0} q^t(i,r)|x_{i,j}| + \sum_{i:y_i=r, x_{i,j}<0} (1-q^t(i,y_i))|x_{i,j}| \ ,$$

the update to row j of W^t to be $\overline{\boldsymbol{w}}_j^{t+1} = \overline{\boldsymbol{w}}_j^t + \boldsymbol{\delta}_j^t$, and let $\boldsymbol{a}$ satisfy $\max_i \left\{\sum_j a_j |x_{i,j}|\right\} \leq \frac{1}{2}$. The change in the multiclass loss, $\Delta_t = L(W^{t+1}) - L(W^t)$, is bounded by

$$\sum_{j=1}^n a_j \sum_{r=1}^k \left(\mu^+_{r,j}(1 - e^{-\delta^t_{j,r}/a_j}) + \mu^-_{r,j}(1 - e^{\delta^t_{j,r}/a_j})\right) \quad .$$

The boosting bounds we derived for the multitask and multiclass losses are syntactically identical, differing only in their computation of the importance weights and correlations. These similarities allow us to attack the update for both problems together, deriving one efficient algorithm for ℓ_1/ℓ_∞-regularized boosting based on a corollary to Thm. 1.

Adding ℓ_∞-regularization terms to the multiclass and multitask losses, we have that the change in the objective is $\Delta_t - \lambda \sum_{j=1}^n \left\|\overline{\boldsymbol{w}}_j^t + \boldsymbol{\delta}_j\right\|_\infty + \lambda \sum_{j=1}^n \left\|\overline{\boldsymbol{w}}_j^t\right\|_\infty$. For simplicity of our derivation, we focus on updating a single row j in W, and we temporarily assume that $\overline{\boldsymbol{w}}_j^t = \mathbf{0}$. We make the substitution $a_j d_r = \delta_{j,r}$. The update to $\overline{\boldsymbol{w}}_j$ is now given by the solution to the following minimization problem:

$$\min_{\boldsymbol{d}} \sum_{r=1}^k \mu_r^+ e^{-d_r} + \mu_r^- e^{d_r} + \lambda \|\boldsymbol{d}\|_\infty \quad . \tag{6}$$

First, we note that the objective function of Eq. (6) is separable in d_j with an ℓ_∞-regularization term. Second, the derivative of each of the terms, dropping the regularizer, is $-\mu_r^+ e^{-d_r} + \mu_r^- e^{d_r}$. Third, the unregularized minimizers are $\tilde{d}_r = \frac{1}{2}\log(\mu_r^+/\mu_r^-)$ (where we allow $\tilde{d}_r = \pm\infty$). We immediately have the following corollary to Thm. 1.

Corollary 4.1. *The minimizing $\boldsymbol{d}^\star$ of Eq. (6) is $\boldsymbol{d}^\star = \mathbf{0}$ if and only if $\sum_{r=1}^k |\mu_r^+ - \mu_r^-| \leq \lambda$. Assume w.l.o.g that $\mu_r^+ \geq \mu_r^-$. When $\boldsymbol{d}^\star \neq \mathbf{0}$, there are sets $B = \{r : |d_r^\star| = \|\boldsymbol{d}^\star\|_\infty\}$ and $U = [k] \setminus B$ such that*

(a) For all $r \in U$, $\mu_r^+ e^{-d_r^\star} - \mu_r^- e^{d_r^\star} = 0$
(b) For all $r \in B$, $|\tilde{d}_r| \geq |d_r^\star| = \|\boldsymbol{d}^\star\|_\infty$
(c) $\sum_{r \in B} \mu_r^+ e^{-\|\boldsymbol{d}^\star\|_\infty} - \mu_r^- e^{\|\boldsymbol{d}^\star\|_\infty} - \lambda = 0$.

Based on the corollary, we can derive an efficient procedure that first sorts the indices in $[k]$ by the magnitude of the unregularized solution $\tilde{d}_r$ (we can assume that $\mu_r^+ \geq \mu_r^-$ and flip signs at the end as in Fig. 1), then solves the following equation for each $\rho \in [k]$:

$$e^{-d} \sum_{r:\tilde{d}_r \geq \tilde{d}_{(\rho)}} \mu_r^+ \ - \ e^d \sum_{r:\tilde{d}_r \geq \tilde{d}_{(\rho)}} \mu_r^- - \lambda = 0 \ . \tag{7}$$

INPUT: Training set $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$
Regularization λ, number of rounds T
Update templates $\mathcal{A} \subseteq \mathbb{R}_+^n$ s.t.
$\forall \boldsymbol{a} \in \mathcal{A}\ \max_i \left\{\sum_{j=1}^n a_j |x_{i,j}|\right\} \leq \frac{1}{2}$
FOR $t = 1$ to T
CHOOSE $j \in \{1, \ldots, n\}$
FOR $i = 1$ to m and $r = 1$ to k
SET $q^t(i,r) = \frac{\exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i)}{\sum_l \exp(\boldsymbol{w}_l \cdot \boldsymbol{x}_i)}$
FOR $r = 1$ to k
$\mu_{r,j}^+ = \sum_{i:y_i=r, x_{i,j}>0} (1 - q^t(i,y_i))|x_{i,j}| + \sum_{i:y_i \neq r, x_{i,j}<0} q^t(i,r)|x_{i,j}|$
$\mu_{r,j}^- = \sum_{i:y_i=r, x_{i,j}<0} (1 - q^t(i,y_i))|x_{i,j}| + \sum_{i:y_i \neq r, x_{i,j}>0} q^t(i,r)|x_{i,j}|$
MINIMIZE for $\boldsymbol{\delta}_j \in \mathbb{R}^k$ such that $\boldsymbol{a}_j \neq 0$ (use Alg. 1)
$\sum_{j,r} a_j \left[\mu_{r,j}^+ e^{-\delta_{j,r}/a_j} + \mu_{r,j}^- e^{\delta_{j,r}/a_j}\right] + \lambda \left\|\overline{\boldsymbol{w}}_j^t + a_j \boldsymbol{\delta}_j\right\|_\infty$
UPDATE
$W^{t+1} = W^t + [\boldsymbol{\delta}_1 \cdots \boldsymbol{\delta}_n]^\top$

Figure 2. AdaBoost for ℓ_1/ℓ_∞-regularized multiclass.

This process continues until we find an index ρ such that the solution $d^\star$ of Eq. (7) satisfies $d^\star \geq \tilde{d}_{(\rho+1)}$, where $\tilde{d}_{(\rho)}$ is the ρ^{th} largest unregularized solution. To develop an efficient algorithm, define $M_\rho^\pm = \sum_{r:\tilde{d}_r \geq \tilde{d}_{(\rho)}} \mu_r^\pm$. To solve Eq. (7) for each ρ, applying the reasoning for Eq. (4) gives

$$d^\star = \log \frac{-\lambda + \sqrt{\lambda^2 + 4M_\rho^+ M_\rho^-}}{2M_\rho^-} \ . \tag{8}$$

When $M_\rho^- = 0$, we get $d^\star = \log(\lambda / M_\rho^+)$. We can use Eq. (8) successively in the algorithm of Fig. 1 by setting $M_{\rho+1}^\pm = M_\rho^\pm + \mu_{(\rho)}^\pm$. To recap, by sorting $\tilde{d}_r = \frac{1}{2}\log(\mu_r^+/\mu_r^-)$ and incrementally updating $M_\rho^\pm$, we can use the algorithm of Fig. 1 to solve the extensions of AdaBoost with ℓ_1/ℓ_∞-regularization in time $O(k \log k)$.

It remains to show how to solve the more general update when $\boldsymbol{w} \neq \boldsymbol{0}$. In particular, we would like to find the minimum of

$$a \sum_{r=1}^k \left(\mu_r^+ e^{-d_r} + \mu_r^- e^{d_r}\right) + \lambda \left\|\boldsymbol{w} + a\boldsymbol{d}\right\|_\infty \ . \tag{9}$$

We can make the transformation $\gamma_r = w_r/a + d_r$, which reduces our problem to the problem of finding the minimizer of $\sum_{r=1}^k \left(\mu_r^+ e^{w_r/a} e^{-\gamma_r} + \mu_r^- e^{-w_r/a} e^{\gamma_r}\right) + \lambda \left\|\boldsymbol{\gamma}\right\|_\infty$ with respect to $\boldsymbol{\gamma}$. This minimization problem can be solved by using the same sorting-based approach, then recovering $\boldsymbol{d}^\star = \boldsymbol{\gamma}^\star - \boldsymbol{w}/a$.

Combining our reasoning for the multiclass and multitask losses, we obtain an algorithm that solves both problems by appropriately setting $\mu_{r,j}^\pm$ and using the algorithm of Fig. 1 to update rows of W. As they are so similar, we present the algorithm only for the multiclass loss in Fig. 2.

5. GradBoost with ℓ_1 & ℓ_1/ℓ_2 Regularization

In this section we shift our attention to a lesser used approach and derive additive updates for the logistic-loss with quadratic upper bounds based on those of Dekel et al. (2005). Concretely, we use bounds of the form

$$L(\boldsymbol{w} + \delta \boldsymbol{e}_j) \leq L(\boldsymbol{w}) + \nabla L(\boldsymbol{w}) \cdot \boldsymbol{e}_j \delta + \frac{1}{2} \delta \boldsymbol{e}_j \cdot D \boldsymbol{e}_j \delta,$$

where D upper bounds $\nabla^2 L(\boldsymbol{w})$ (in the binary case, $D = \mathrm{diag}(1/4 \sum_{i=1}^m x_{i,j}^2)$). We term these methods GradBoost for their use of gradients and bounds on the Hessian. We make no claim about whether the resulting algorithms entertain the weak-to-strong learnability properties of AdaBoost. The quadratic bounds allow us to perform boosting-style steps with ℓ_2 and ℓ_2^2 regularization in addition to the regularizers studied above. We start with the binary logistic loss. GradBoost, similar to AdaBoost, can use a template vector $\boldsymbol{a}$ to parameterize updates. We focus on single-coordinate updates for cleanliness, however. The following lemma gives a progress bound for GradBoost.

Lemma 5.1 (GradBoost Progress Bound). *Let $\boldsymbol{g}$ denote the gradient of the logistic loss, $g_j = -\mu_j^+ + \mu_j^-$. Let $a_j = 1/\sum_i x_{i,j}^2$ and $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \delta^t \boldsymbol{e}_j$ with $\delta^t = a_j d^t$. Then the change, $\Delta_t = L(\boldsymbol{w}^t) - L(\boldsymbol{w}^{t+1})$, in the logistic-loss between iterations of GradBoost is lower bounded by*

$$\Delta_t \geq -a_j \left(g_j d^t + \frac{(d^t)^2}{8}\right) \ = \ - \left(g_j \delta^t + \frac{(\delta^t)^2}{8 a_j}\right) \ .$$

To derive a usable bound for GradBoost with ℓ_1-regularization, we replace the progress bound in lemma 5.1 with a bound dependent on $\boldsymbol{w}^{t+1}$ and $\boldsymbol{w}^t$ by substituting $\boldsymbol{w}^{t+1} - \boldsymbol{w}^t$ for $\boldsymbol{\delta}^t$. Incorporating ℓ_1-regularization, we get that $Q(\boldsymbol{w}^{t+1}) - Q(\boldsymbol{w}^t)$ is upper bounded by

$$g_j(w_j^{t+1} - w_j^t) + \frac{1}{8a_j}(w_j^{t+1} - w_j^t)^2 + \lambda|w_j^{t+1}| - \lambda|w_j^t| \tag{10}$$

The bound above is separable, so Thm. 1 gives the update

$$w_j^{t+1} = \mathrm{sign}(w_j^t - 4a_j g_j) \left[|w_j^t - 4a_j g_j| - 4a_j\lambda\right]_+ \tag{11}$$

We can use Eq. (11) to derive a GradBoost algorithm for the ℓ_1-regularized logistic loss. The algorithm is omitted as we give a general multiclass version in the sequel. It is also possible to use Thm. 1 to obtain new coordinate descent methods for losses with ℓ_1/ℓ_∞ regularization. Due to lack of space we omit details but report the performance of these variants in Sec. 6.

One form of regularization that has not been considered in the standard boosting literature is ℓ_2 or ℓ_2^2 regularization. The lack thereof is a consequence of AdaBoost's exponential bounds on the loss. GradBoost, however, can straightforwardly incorporate ℓ_2-based penalties, since it instead uses linear and quadratic bounds on the decrease in the loss. We focus on multiclass GradBoost, as modifications for multitask follow the lines of derivation discussed

INPUT: Training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$;
Regularization λ; number of rounds T

FOR $t = 1$ to T

CHOOSE $j \in \{1, \ldots, n\}$

SET $a_j = 1/\sum_i x_{i,j}^2$

FOR $i = 1$ to m and $r = 1$ to k

// Compute importance weights for each class

SET $q^t(i,r) = \frac{\exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i)}{\sum_{l=1}^k \exp(\boldsymbol{w}_l \cdot \boldsymbol{x}_i)}$

FOR $r = 1$ to k

// Compute gradient terms

SET $g_{r,j} = \sum_{i=1}^m (q^t(i,r) - \mathbf{1}\{r = y_i\}) x_{i,j}$

$\boldsymbol{g}_j = [g_{1,j} \cdots g_{k,j}]$

$\overline{\boldsymbol{w}}_j^{t+1} = \left(\overline{\boldsymbol{w}}_j^t - 2a_j \boldsymbol{g}_j\right) \left[1 - \frac{2a_j\lambda}{\left\|\overline{\boldsymbol{w}}_j^t - 2a_j \boldsymbol{g}_j\right\|_2}\right]_+$

Figure 3. GradBoost for ℓ_1/ℓ_2-regularized multiclass.

thus far. We focus particularly on mixed-norm ℓ_1/ℓ_2-regularization (Obozinski et al., 2007), in which rows from the matrix $W = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_k]$ are regularized together in an ℓ_2-norm. This leads to the following modification of the multiclass objective $Q(W)$ from Eq. (2):

$$\sum_{i=1}^m \log\left(1 + \sum_{r \neq y_i} e^{\boldsymbol{w}_r \cdot \boldsymbol{x}_i - \boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i}\right) + \lambda \sum_{j=1}^n \|\overline{\boldsymbol{w}}_j\|_2 \quad (12)$$

Generalizing the GradBoost progress bound while using the normalization $a_j = 1/\sum_i x_{i,j}^2$ as before (omitting details), we upper bound $Q(W^{t+1}) - Q(W^t)$ by

$$\sum_{r=1}^k \left(g_{r,j} - \frac{w_{j,r}^t}{2a_j}\right) w_{j,r}^{t+1} + \frac{1}{4}\sum_{r=1}^k \frac{(w_{j,r}^{t+1})^2}{a_j} \quad (13)$$
$$+ \sum_{r=1}^k \left(\frac{(w_{j,r}^t)^2}{4a_j} - g_{r,j} w_{j,r}^t\right) + \lambda\left(\left\|\overline{\boldsymbol{w}}_j^{t+1}\right\|_2 - \left\|\overline{\boldsymbol{w}}_j^t\right\|_2\right)$$

where $g_{r,j}$ is the derivative of the multiclass loss w.r.t the j^{th} weight for class r. The above bound is evidently a separable quadratic function with ℓ_2-regularization. We would like to use Eq. (13) to perform block coordinate descent on the ℓ_1/ℓ_2-regularized loss Q from Eq. (12). Defining the gradient vector $\boldsymbol{g}_j = [g_{1,j} \cdots g_{k,j}]^\top$ and using basic properties from convex analysis, we obtain that the update performed to minimize Eq. (13) with respect to row $\overline{\boldsymbol{w}}_j$ is

$$\overline{\boldsymbol{w}}_j^{t+1} = \left(\overline{\boldsymbol{w}}_j - 2a_j \boldsymbol{g}_j\right) \left[1 - \frac{2a_j\lambda}{\left\|\overline{\boldsymbol{w}}_j^t - 2a_j \boldsymbol{g}_j\right\|_2}\right]_+ \quad (14)$$

To recap, we obtain an algorithm for minimizing the ℓ_1/ℓ_2-regularized multiclass loss by iteratively choosing indices j and then applying the update of Eq. (14). The algorithm is given in Fig. 3.

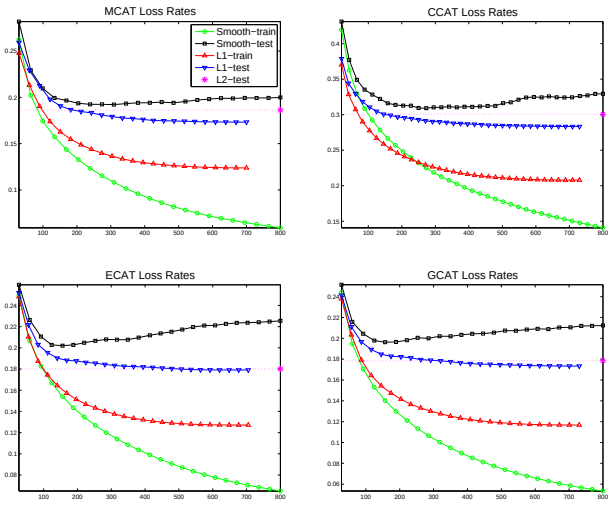

Figure 4. Results on the 4 top-level classes from RCV1.

6. Experiments

One of the objectives of this section is to demonstrate empirically that the proposed algorithms are effective in obtaining sparse and accurate models. In our first series of experiments, we focus on boosting and feature induction, investigating the effect of ℓ_1-regularization and its ability to automatically cease inducing new features in the presence of regularization. We examined both classification and regression tasks. For classification, we used the Reuters RCV1 Corpus (Lewis et al., 2004), which consists of $804,414$ news articles and around 100,000 words that constitute our features. Each article is labeled by at least one label from the set MCAT, CCAT, ECAT, and GCAT, and we train a classifier for each each class. We show average logistic loss rates over a series of tests using 30,000 randomly chosen articles with a 70/30 training/test split in Fig. 4 (error rates are similar and we omit them for space). As a baseline for comparison, we used boosting regularized with a smooth-ℓ_1 penalty (Dekel et al., 2005), an approximation to the ℓ_1-norm. We also compared our approach to ℓ_2-regularized logistic regression with features chosen using mutual information with the target (details omitted). The regularization parameters were chosen using 5-fold cross validation. For both boosting algorithms, we ran the "totally corrective" variant (Warmuth et al., 2006). Concretely, we added the 30 top-scoring features on every iteration for the ℓ_1 booster and the single top-scoring feature for the smooth-ℓ_1 regularized booster, then reoptimized the weights of all the features previously selected. The graphs in Fig. 4 suggest an interesting story. In all of them, the ℓ_1-regularized booster actually ceased adding features around iteration 30, including about 700 features with non-zero weights after the back-pruning/optimization steps. Therefore, the plots for ℓ_1-AdaBoost terminate early, while the smooth-ℓ_1 booster keeps adding features and starts overfitting to the training set as early as iteration 200.

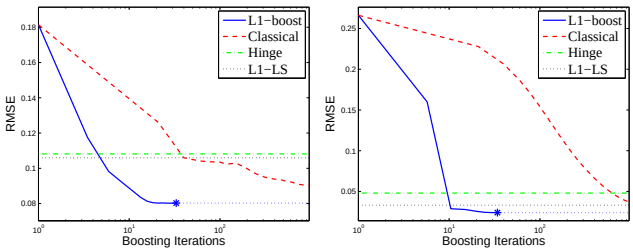

Figure 5. Results for Boston Housing (left) and Aileron.

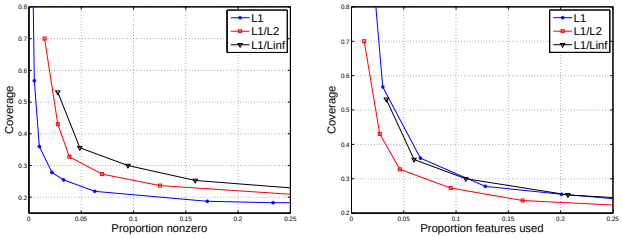

Figure 6. Left: test set coverage versus overall sparsity of W. Right: test set coverage versus row sparsity of W.

We also performed comparisons on regression tasks. We describe here results for the Boston housing data set from the UCI repository and an F16 aircraft control dataset, where the goal is to predict an action on the ailerons of the aircraft given its state. We used the ε-insensitive regression loss (Dekel et al., 2005) to learn a predictor $\boldsymbol{w}$. In this case, our objective is $\sum_{i=1}^{m} \log\left(1 + e^{\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i - \varepsilon}\right) + \log\left(1 + e^{y_i - \boldsymbol{w}\cdot\boldsymbol{x}_i - \varepsilon}\right)$, which approximates ε-insensitive hinge regression. For ε-insensitive regression, an analysis similar to that for standard boosting can be performed to compute μ^+ and μ^- for every feature, which allows us to perform boosting as already described. For these tests, we compared unregularized "classical" (forward-greedy) sequential AdaBoost, our ℓ_1-regularized totally-corrective boosting, ℓ_1-regularized least squares (Friedman et al., 2007), and ℓ_2-regularized ε-insensitive hinge loss. The boosters used a countably infinite set of features by examining all products of features and were started with a single bias feature. The algorithms could thus construct arbitrarily many products of raw features as base hypotheses and explore correlations between the features. For the ℓ_1-regularized least squares and the hinge loss, we simply trained on the base regressors.

Fig. 5 shows the results for these tests with the Housing results on the left. Each plot contains the root-mean-square error on test (the absolute error on test is qualitatively similar). The ℓ_1-regularized booster stopped after inducing an average of under 35 features, marked with a star in the graphs, after which a dotted line indicates the booster's intact performance. We see that even when classical AdaBoost is allowed to run for 1,000 iterations, its performance on test does not meet the performance for the 35 features induced by the ℓ_1-regularized model. In fact, even after 3,000 iterations, the smooth-ℓ_1 variant was not able to outperform the 35 feature model built by the ℓ_1-penalized version. Furthermore, the latter trains at least an order of magnitude faster than the classical forward greedy regressor and results in a significantly simpler model. ℓ_1-penalized AdaBoost also outperforms ℓ_1-penalized least squares and the ℓ_2-regularized hinge loss with respect to both the squared and absolute errors.

In the next set of experiments, we compare the different structured regularizers with multiclass logistic losses to unstructured regularization. As our datasets are single-label, we omit experiments with multitask losses. For all multiclass experiments, we focus on two error metrics. The first is misclassification rate. The second is *coverage*, which measures how wrong a classifier is. Given k weight vectors $\boldsymbol{w}_r$ and an example $\boldsymbol{x}_i$ with label y_i, the coverage is the correct weight vector's position in the sorted list of inner products $\boldsymbol{w}_r \cdot \boldsymbol{x}_i$. For example, if $\boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i$ is the largest, the coverage is 0, if it is third, the coverage is 2.

We begin with the StatLog LandSat dataset (Spiegelhalter & Taylor, 1994), which consists of spectral values of pixels in 3×3 neighborhoods in a satellite image. We expanded the data by taking products of all features, giving 1296 features per example. The goal is to classify a pixel (a piece of ground) as one of six ground types. In Fig. 6, we plot coverage as a function of sparsity and as a function of the number of features actually used (trained with GradBoost). The plot on the left shows the test set coverage as a function of the proportion of zeros in the learned weight matrix W (we plot results for a training set of 240 examples per class as results are similar for smaller and larger sets). On the right, we show test set coverage as a function of the actual number of features that need to be computed to classify a piece of ground—that is, the proportion of *all zero* rows in W. From the plots, we see that for a given performance level, the ℓ_1-regularized solution is sparser in terms of the absolute number of zeros. However, the ℓ_1-regularized classifier requires at least 50% more features be computed than does the ℓ_1/ℓ_2-regularized classifer for the same test accuracy. The results for misclassification rates are similar, and the variance for each point in the plots is smaller than 10^{-3}.

We also ran tests using the MNIST handwritten digits database. The dataset consists of 60,000 training examples with a 10,000 example test set and has 10 classes. Each image is a gray-scale 28×28 image, which we represent as $\boldsymbol{x}_i \in \mathbb{R}^{784}$. Rather than directly use the input $\boldsymbol{x}_i$, however, we learned weights $\boldsymbol{w}_j$ using Kernel-based weak hypotheses $h_j(\boldsymbol{x}) = K(\boldsymbol{x}_j, \boldsymbol{x})$, $K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|^2}$ for $j \in S$, where S is a 2766 element support set we generate by doing one pass through the data with the perceptron and keeping examples on which it makes mistakes. This gives a 27,660 dimensional multiclass problem. On the left side of Fig. 7, we plot the coverage rate for each algorithm on the

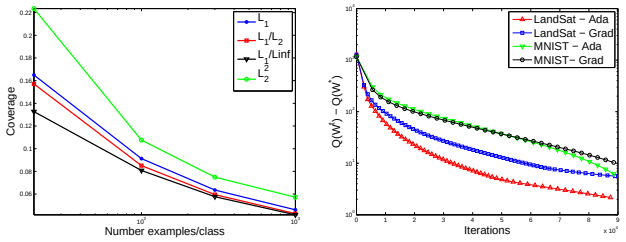

Figure 7. Left: Coverage on MNIST. Right: Training time

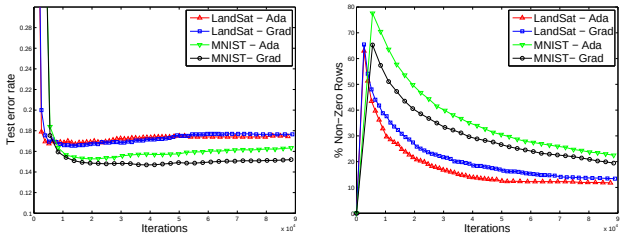

Figure 8. Left: MNIST/LandSat error rate. Right: Sparsity.

10,000 example test set versus the number of training examples used per class (we choose regularization values by cross-validation). Each improves as the number of training examples grows; however, it is clear that the sparsity inducing regularizers, specifically the structural ℓ_1/ℓ_∞ and ℓ_1/ℓ_2 regularizers, give better performance than the others. The error rate on the test set is roughly 50% the coverage and qualitatively similar.

We conclude with a direct comparison of AdaBoost and GradBoost with ℓ_1/ℓ_∞-regularization. On the left side of Fig. 7 and in Fig. 8, we plot the training objective, test error rate, and sparsity of the classifiers as a function of training time for both AdaBoost and GradBoost on the LandSat dataset and MNIST dataset. From Fig. 8, we see that both AdaBoost and GradBoost indeed leverage induction during the first few thousand iterations, adding many features that contribute to decreasing the loss. They then switch to a backpruning phase in which they remove features that are not predictive enough without increasing mistakes on the test set. We saw similar behavior across many datasets, which underscores the ability of the algorithms to perform both feature induction and backward pruning in tandem.

Though this paper omits the following, we note that our algorithms all enjoy convergence guarantees, provide scoring mechanisms for induction of features in boosting, and give termination criteria for boosting based on the sparsifying regularizers. These results and full proofs are available in the long version of the paper on the the first author's website: http://www.cs.berkeley.edu/˜jduchi.

References

Bertsekas, D. (1999). *Nonlinear programming*. Athena Scientific.

Collins, M., Schapire, R., & Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, *47*, 253–285.

Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2005). Smooth epsilon-insensitive regression by loss symmetrization. *J. Machine Learning Research*, *6*, 711–741.

Dudík, M., Phillips, S. J., & Schapire, R. E. (2007). Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *J. Machine Learning Research*, *8*, 1217–1260.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, *28*, 337–374.

Friedman, J., Hastie, T., & Tibshirani, R. (2007). Pathwise coordinate optimization. *Annals of Applied Statistics*, *1*, 302–332.

Koh, K., Kim, S., & Boyd, S. (2007). An interior-point method for large-scale ℓ_1-regularized logistic regression. *J. Machine Learning Research*, *8*, 1519–1555.

Lewis, D., Yang, Y., Rose, T., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *J. Machine Learning Research*, *5*, 361–397.

Meir, R., & Rätsch, G. (2003). An introduction to boosting and leveraging. *Advanced Lectures on Machine Learning* (pp. 119–184). Springer.

Negahban, S., & Wainwright, M. (2008). Phase transitions for high-dimensional joint support recovery. *Advances in Neural Information Processing Systems 22*.

Obozinski, G., Taskar, B., & Jordan, M. (2007). *Joint covariate selection for grouped classification* (Technical Report 743). Dept. of Statistics, University of California Berkeley.

Spiegelhalter, D., & Taylor, C. (1994). *Machine learning, neural and statistical classification*. Ellis Horwood.

Warmuth, M., Liao, J., & Ratsch, G. (2006). Totally corrective boosting algorithms that maximize the margin. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 1001–1008).

Zhang, T. (2008). Adaptive forward-backward greedy algorithm for sparse learning with linear models. *Advances in Neural Information Processing Systems 22*.

Zhang, T., & Yu, B. (2005). Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, *33*, 1538–1579.

Zhao, P., & Yu, B. (2006). On model selection consistency of Lasso. *J. Machine Learning Research*, *7*, 2541–2567.